JACOBS
UNIVERSITY

Automation of a Linear Gradient Mixer for Chromatography

by

Kelan F. Garcia Osorio

Bachelor Thesis in Electrical and Computer Engineering

Ph.D. Fangning Hu

Name and title of the supervisor

Date of Submission: May 16, 2021

Jacobs University — Electrical and Computer Engineering

Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung. Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

Signature Bremen, Date

Acknowledgment

First, I will like to thank Prof. Dr. Detlef Gabel for introducing me to this area of research and let me work with him. Then, I will like to thank my supervisor, Ph.d. Fangning Hu for all her guidance and instruction during the whole time of my work. Her trust, dedication, availability and support was key for the success of this thesis. I will love to thank my parents for supporting and trusting me throughout all my studies. Finally, I will like to thank my siblings for always being there for me and cheering me up whenever I need it.

Bremen, Kelan Garcia
May 16, 2021

Abstract

This thesis investigates on how to create a linear gradient mixer machine. The machine automates the process that designs the solvent needed in the mobile phase of chromatography gradient. The purpose of the investigation of this machine, is to design a linear gradient mixer that is cheap to build, that it can be use at any lab without special equipment or license, and the user can design the needed linear gradient. The final machine is design for long linear gradient processes, but with a maximum of 6 hours. The actual linear increment done by the gradient mixer was measured and obtained by collecting several samples during the process and doing a proton nuclear magnetic resonance analysis to them. The linear increment graph follows the behavior of the theory graph but it is not strictly tight to it. Nevertheless, it is tight enough for its purpose.

Contents

Chapter 1

Introduction

Chromatography is a technique that is used for the separation of chemical substances from a mixture. It was first devised in 1900 by Mikhail Tsvet, the Russian-Italian scientist [1] for separation of plant pigments, in $1930s$ and $1940s$ new chromatography techniques were developed for many separation processes. The process consists of a mobile phase and a stationary phase. Stationary phase is the fluid mixture that wants to be separated into its components; this solution is inside a fixed system (a column, a capillary tube, a plate, or a sheet). The mobile phase is the solvent (input mixture) that is carried through the stationary phase (column, capillary tube, etc...) and separates the mixture into its components. The composition of the solvent in the mobile phase depends on the components of the stationary phase that has to be separated. This composition varies with time so it can be able to separate all the components.

In chromatography gradient, the solvent in the mobile phase is composed by two substances, a polar and a non-polar substance. The percentage of each substance has to vary linearly with time, e.g., it starts with a 100% solvent of non-polar and 0% of polar then with time it starts to slowly decreasing the percentage of the non-polar and the polar one starts to slowly linearly increase (90% non-polar, 10% polar). This linear change in elution strength from weak (non-polar) to strong (polar) is called a linear gradient mixer [2].

Currently, on a normal lab this procedure is done manually. Two beakers are used, one is filled with a known amount of polar chemical and the other one is filled with a known amount of the non-polar chemical. First, a tube is connected from the non-polar beaker and the fluid is pumped into the stationary phase, then another tube is added and this one is connected between the polar beaker and the non-polar beaker. Then, the speed of the pump needs to be manually adjusted so it can achieve the desired linear gradient. This adjust depends on the change of the amount of fluid on each beaker and by calculating the percentages. As it can be seen, this manual procedure is complicated, not reliable and prompt to errors. There are other professional gradient mixers on the market, but they are expensive and some require a special medical license to have them. This thesis investigates on how to automate a linear gradient mixer that is able to make the solvent needed in that specific time in the mobile phase with all the parameters given. One that is cheap, reliable, and can be implemented in any laboratory without external equipment.

In order to create a circuit that can automate the process of a linear gradient needed in a chromatography, the circuit needs to be able to decide when is the ideal time to change the state between the polar chemical and the non-polar chemical of the solenoid valve so it can achieve the desired linear gradient given by the user with all the given restrictions. For this, the circuit consists of a microcontroller, together with a solenoid valve, a constant flow rate pump, a $433MHz$ wireless antenna so it can control the pump, the use of transistors acting as amplifier switches and Op-amps for the solenoid valve are investigated, a TFT LCD touch screen to show the results and also to input the needed parameters. All of these need to be integrated into a single final product. Then a software is designed in a way that it can control all the hardware and can achieve the expected linear gradient. It also has to be able to pause and play the linear gradient process at any time.

The chapter overview of this thesis is the following: In Chapter 2, an overview of the wanted linear gradient mixer is given. In Chapter 3, each component from the gradient mixer is designed in this chapter. In Chapter 4, the results of the designed experiment for testing the gradient mixers are given. Also, the final products are shown as the 3D case and the software interface.

Chapter 2

Linear Gradient Mixer Overview

The linear gradient mixer behavior and requirements, in order to achieve the aim, are the following:

1. Everything needs to be controlled by one thing.
2. The gradient mixer machine should be able to control the pump wirelessly.
3. It needs to be able to switch between the chemicals (the polar and non-polar solvents).
4. The user needs to be able to manually insert the 4 needed parameters $(t_1, A\%, t_2, B\%)$. Based on these values the gradient mixer will be able to calculate all needed flow times for each solvent, so it can perform the wanted linear gradient mixer. Automating the whole process.
5. $A\%$ is the instantaneous percentage with which the polar components starts, this will last for t_1 minutes. After t_1, the instantaneous percentage of the polar chemical starts to linearly increase and the instantaneous percentage of the non-polar chemical starts to linearly decrease. The linear increase should continue until t_2, at that time the instantaneous percentage should be $B\%$. Figure 2.1 gives the graph representation, also in the description the restrictions for $(t_1, A\%, t_2, B\%)$ are given.
6. The instantaneous percentage depends on the instantaneous time. A $6000ms$ bandwidth window is given. The polar percentage depends on the time the polar solvent flow in that $6000ms$ window (e.g. if $A\% = 20\%$ then, the time the polar solvent needs to flow is $6000ms \cdot 20\% = 1200ms$; and the non-polar solvent flow time is $6000ms \cdot 80\% = 4800ms$). Then, in the next $6000ms$ window, the polar time is increased depending on the percentage increase needed $\Delta\%$(e.g. in the next window the polar percentage is $A\% + \Delta\%$ therefore, the polar time flow is $6000ms \cdot (A\% + \Delta\%)$; and the non-polar time flow is $6000ms \cdot (100 - (A\% + \Delta\%))$.
7. The linear gradient mixer should have a limit maximum time of operation of 6 hours (360 min).

8. In case of emergency, the gradient mixer can stop all the process with the press of an interrupt button and give visual signals with LEDs.
9. The final product circuit needs to be protected from damages caused by small liquid leaking from the valve, the tubes or the pump.

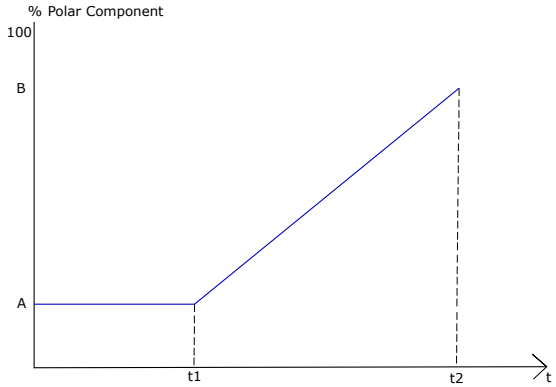


Figure 2.1: Polar Component, $t_1 \leq 10; A\% < B\%; t_1 < t_2; B \leq 100$;and $t_2 \leq 360min$

Based on the requirements given, the selected components are the following:

1. For a micro-controller an Arduino Mega 2560 is used,
2. 3.2" TFT LCD Touch Screen: All user parameters are inserted via the touch screen. Also, the user can control all components from here.
3. TFT Mega Shield V2.2: A shield needed to adapt the LCD touch screen to the Arduino.
4. 3/2 Way $12V$ Solenoid Valve: it is used so the linear gradient mixer can change between the two solvents the polar and the non-polar. Since, it is a $12V$ valve a solenoid valve controller circuit needs to be designed so it can transform the $5V$ square signal of the ATMEGA2560 on the Arduino Mega into a $12V$ square signal for the valve.
5. $433Mhz$ Antennas (transmitter and receiver): the receiver is used to collect and decode the signals that are used to control wirelessly the $433Mhz$ power outlet used in the pump.The transmitter is used to replicate that sequence when it is needed.
6. Two push buttons and one LED: The buttons are needed to activate and deactivate the emergency stop protocol. The LED indicates that the mixer is in stop.
7. Plastic filament, to be able to print a 3D case that is going to be designed to enclose and protect all the circuit.

The brain of this linear gradient mixer machine is the ATMEGA2560 on the Arduino Mega 2560. It is used to calculate the needed time for each state in the solenoid valve, this time varies in the process; it controls the state of the solenoid valve, controls the pump wirelessly, has an emergency interrupt button that can stop the whole process and it can be controlled by an LCD touch screen.

The block circuit diagram that is made to achieve the given descriptions and requirements, is on Figure 2.5. To protect the final product from liquid leaking from the valve, it will be enclosed on a 3D printed case designed to its size. To start, the circuit for the solenoid valve PCB controller is designed so it can amplify the $5V$ pulse width modulation from the arduino to a $12V$ pulse modulation.

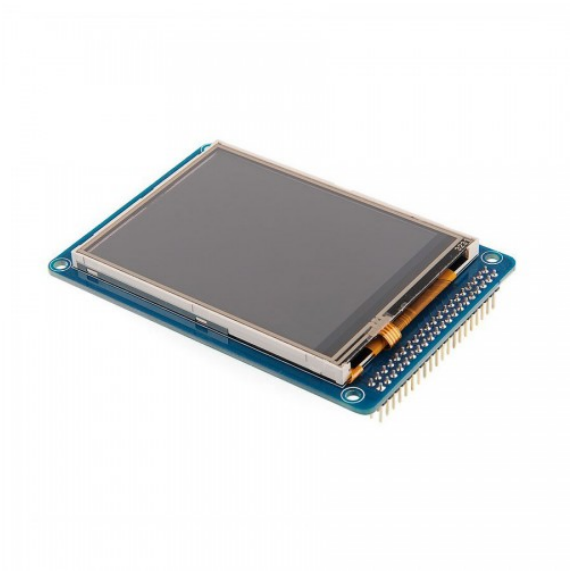


Figure 2.2: 3.2" TFT LCD Touch Screen

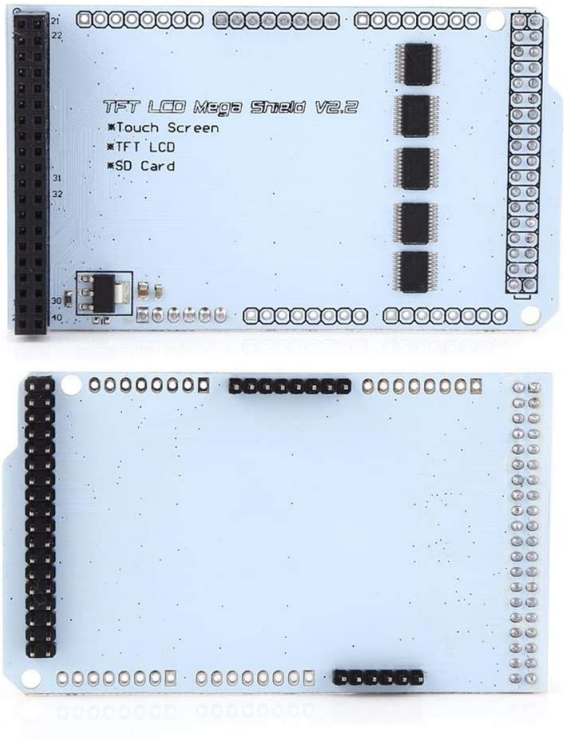


Figure 2.3: TFT Mega Shield V2.2

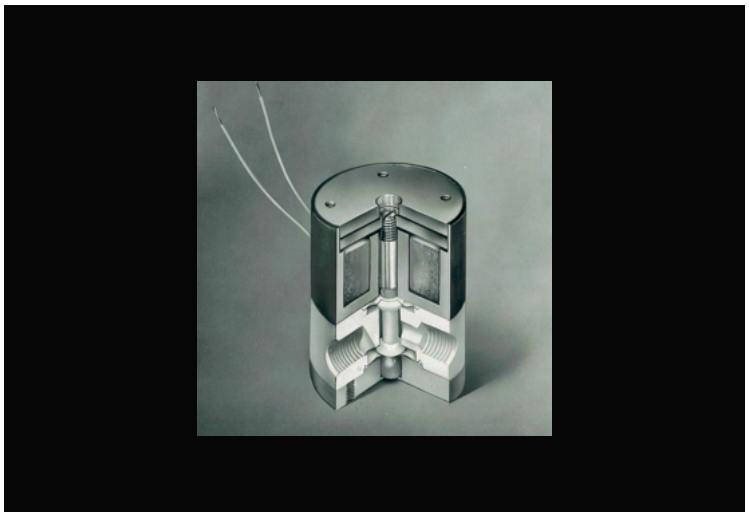


Figure 2.4: 3/2-way NC 12V Solenoid Valve

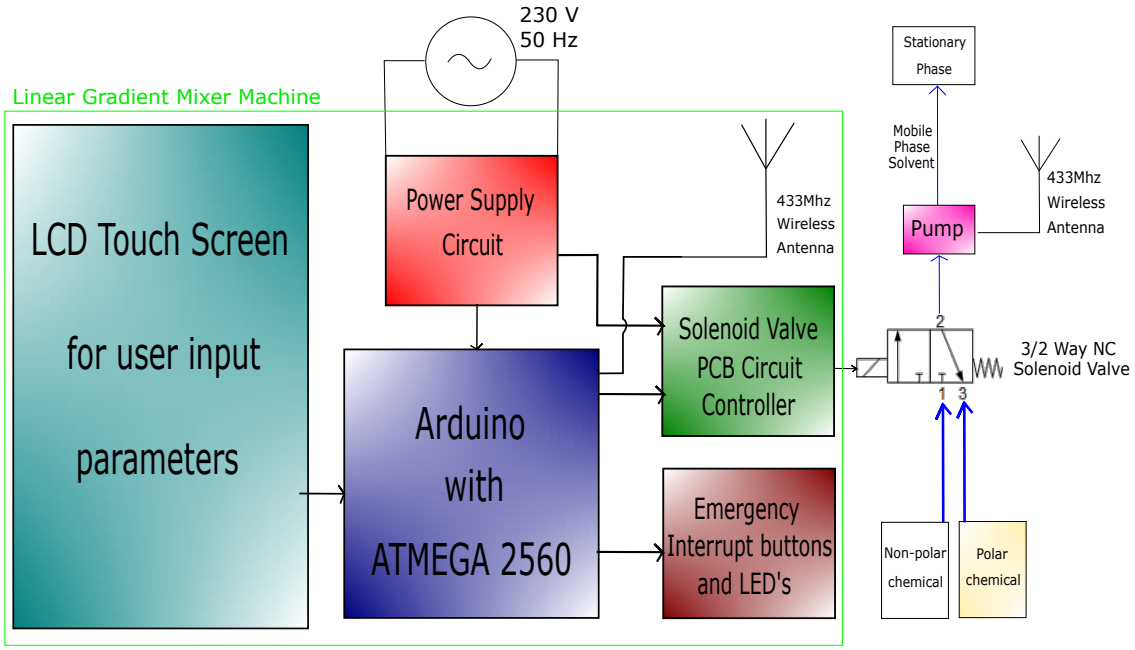


Figure 2.5: Gradient Mixer Block Diagram

Chapter 3

Design

3.1 Solenoid Valve PCB Controller

The Solenoid Valve PCB controller is the circuit that will change the position state of the $12V$ NC $3/2$ Way Solenoid Valve. When the solenoid valve is Off $(0V)$ the valve allows the flow of the non-polar solvent to the pump. When the valve is On $(12V)$, the valve changes state and it now allows the flow of the polar solvent to the pump. This circuit will have an input square pulse signal $x(t)$, with $V_{max} = 5V$, that comes from the ATMEGA 2560. The output signal $y(t)$ should have the same behavior as the input but should have a V_{max} of $12V$ rather than $5V$. Two circuits were made for this purpose. On this chapter both circuits are analyzed and compared. At the end the best one is used.

3.1.1 First Solenoid Valve Controller Circuit Proposal

Since the circuit only needs to amplify the input signal voltage, the Op-amp LM741CN as a non-inverted amplifier was considered. In a non-inverting amplifier, first the gain constant A_{gain} needs to be calculated.

$$v_o = A_{gain} \cdot v_i$$
$$A_{gain} = \frac{v_o}{v_i} = \frac{12V}{5V} = 2.4$$

Formula for a non-inverting amplifier.

$$v_o = \left(1 + \frac{R_1}{R_2}\right) \cdot v_i$$
$$A_{gain} = \left(1 + \frac{R_1}{R_2}\right) = 2.4$$
$$\frac{R_1}{R_2} = 1.4$$
$$R_1 = 14,000\Omega; R_2 = 10,000\Omega$$

Since the solenoid valve uses a magnetic coil to change the state of the valve, it is considered an inductive component. When the magnetic coil loses power it generates transient voltages that can cause damages to other components in the circuit. To protect the circuit from these spikes, a diode is placed at across both contacts, this diode eliminates the transient voltages spikes.[3]

The diode is placed from the negative side of the magnetic coil to the positive side (anode side) of the diode. Since, diodes only allow current to flow in one direction it will protect the spikes to the other direction. The polarity needs to be taken in consideration, if not it will be a dead short between power and ground. A resistor load is added at the output of the Op-amp, in parallel with the diode. Its voltage drop is the output signal $y(t)$.

According to the LM741CN Op-amp data sheet the power supply needs two voltage sources and two $100nF$ capacitors as shown in figure 3.1 [4]

3.1.1.1 Circuit

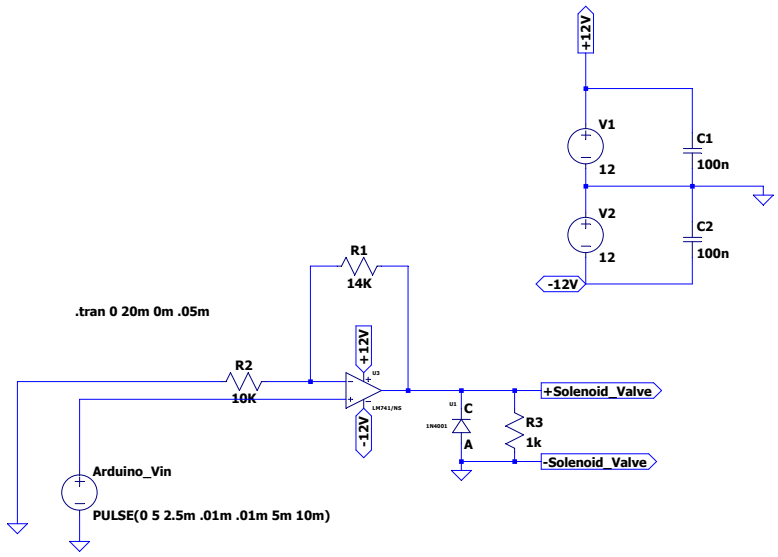


Figure 3.1: 1^{st} Solenoid Valve Controller circuit proposal on LTSpice

3.1.1.2 Circuit Simulation

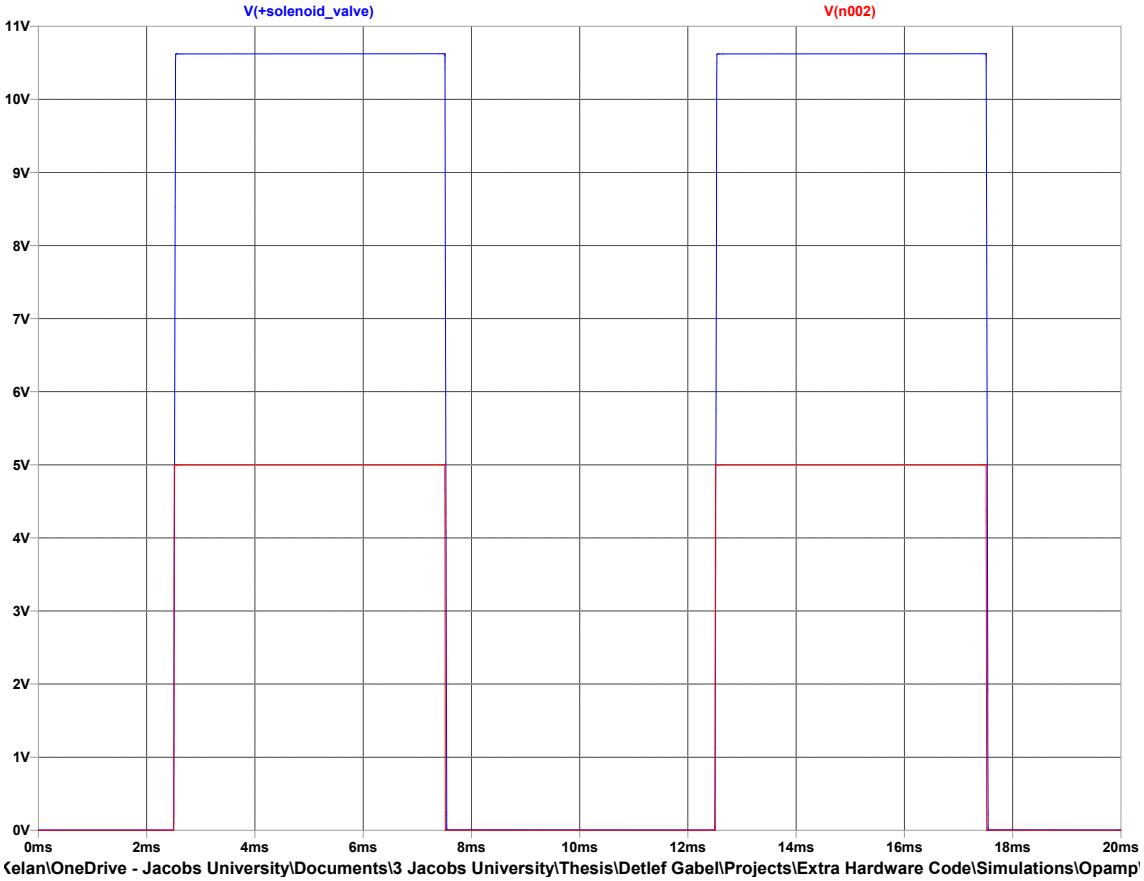
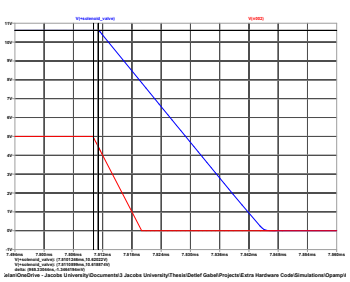
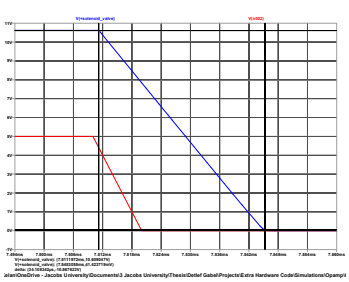


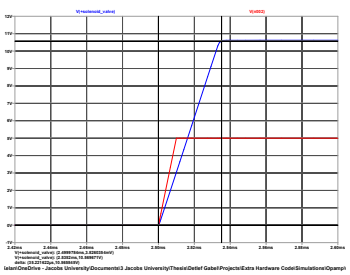
Figure 3.2: Simulation Results; $x\left(t\right)$ = red; $y\left(t\right)$ = blue



(a) $t_{on_{falldelay}} = 965.3ns$



(b) $t_{fall} = 34.1\mu s$



(c) $t_{rise} = 35.2\mu s$; $t_{on_{risedelay}} = 24.9\mu s$

Figure 3.3: Delays

3.1.1.3 Oscilloscope

The circuit was assembled on a breadboard and test it. The results were measured with an oscilloscope. The overall view is on figure 3.4. As it can be seen, the output is not exactly as simulated, it has a $1V$ offset, the V_{max} is not exactly as expected, the rise and fall delays are different.

Figure 3.5a shows the On fall delay time $t_{on_{falldelay}}$ of the output signal compared to the input signal. The $t_{on_{falldelay}}$ is the time the output takes to start decreasing from V_{max} compared to the input signal. Figure 3.5b shows the time it takes the output signal to reaches its minimum. Figure 3.5c gives the time it takes to the output to raise.

$$t_{on_{falldelay}} = 6.60\mu s; t_{off} = 22.2\mu s$$

$$t_{fall} = t_{off} - t_{on_{falldelay}}$$

$$t_{fall} = 22.2\mu s - 6.60\mu s = 15.6\mu s$$

$$t_{rise} = 13.4\mu s$$

Since the arduino will control the solenoid valve in the range of milliseconds, these delays in micro-seconds should not affect the percentages of the gradient mixer. In Section 3.4.1 it is proved that the maximum frequency of $x(t)$ is $100Hz$, that is why the frequency of the experimental input signal is $100Hz$.

Delay Type	Simulation	Experimental
$t_{on_{falldelay}}$	$965.3ns$	$6.60\mu s$
t_{fall}	$34.1\mu s$	$15.6\mu s$
$t_{on_{risedelay}}$	$24.9\mu s$	$13.4\mu s$
t_{rise}	$35.2\mu s$	$13.4\mu s$

Table 3.1: Comparison of the delays simulated and experimental of the Op-amp circuit

Something noted is that the circuit depends on an external factor, temperature. When it first starts working, it controls the solenoid valve as expected. However, if the gradient mixer is working for an extended period, some components of the circuit start to warm up causing the output current to start dropping. The solenoid valve will stop changing its state due to the low current. This problem can be due to the two 12V@1A power supplies in the circuit, rather than just using one power supply. This issue can be solved by reducing the time limit of use of the gradient mixer in one run, so it does not warm up. The disadvantage of this solution is that the new time limit will be small compared to the 6 hours' time limit desired. Another alternative is adding a heat sink to the entire PCB; the disadvantage is that it will make the final product thicker.

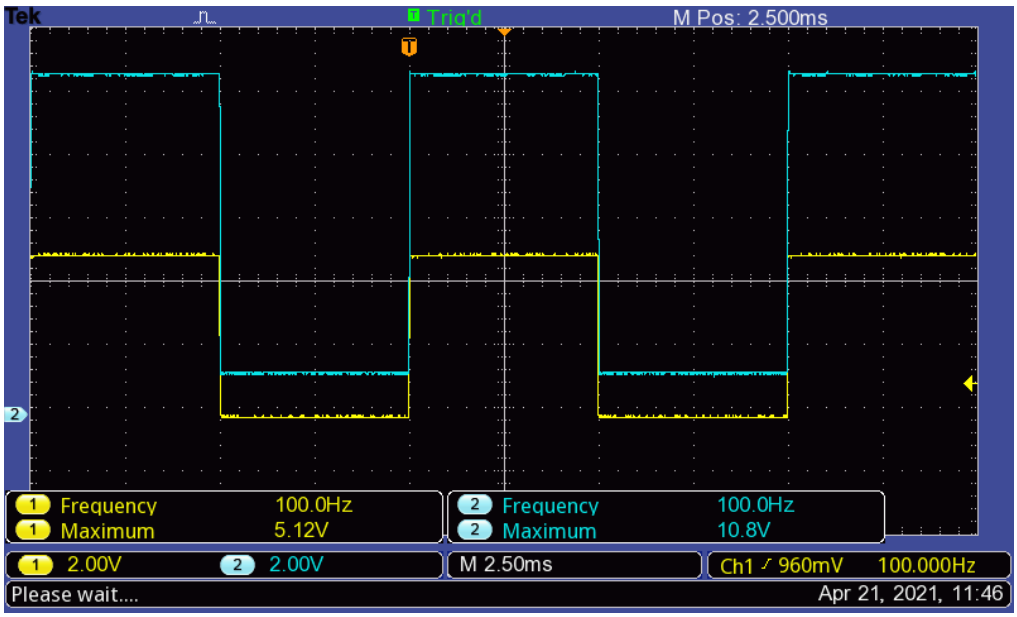
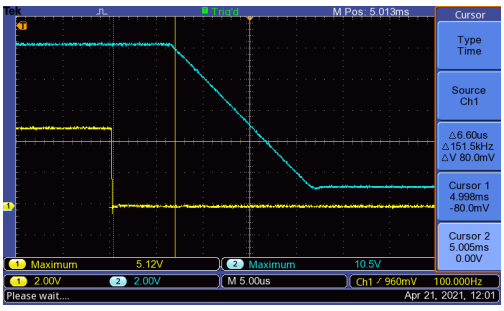
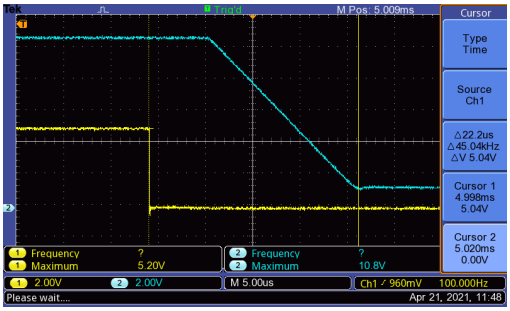


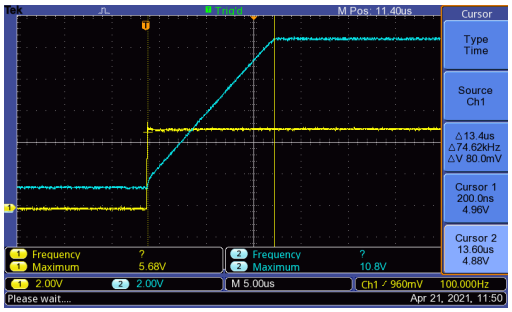
Figure 3.4: Channel 1 (Yellow) = Input Signal; Channel 2 (Blue) = Output Signal



(a) $t_{on_{falldelay}} = 6.60\mu s$



(b) $t_{off} = 22.2\mu s$



(c) $t_{rise} = 13.4\mu s; t_{on_{risedelay}} \approx t_{rise}$

Figure 3.5: Delays

3.1.2 Second Solenoid Valve Controller Circuit proposal

The second proposal is to design a power switch using two npn transistors. Some benefits of using a power transistor as a switch are: they are fast, cheaper than op-amps, their power ratings are from 10 to $30W$, frequency ratings from 1 to $100Mhz$, maximum I_C values range between 1 to $100A$.[5] However, it also has some disadvantages such as: It is more fragile, small amounts of currents are needed, otherwise it can damage the transistor. In this circuit a $12V$ power supply is needed, the same diode and resistor that were used for safety are also used in this circuit. The reason was explained on the previous circuit design. To make a switch transistor, two npn transistors are connected as shown on figure 3.6 The values of R_1, R_2 and D_1 depend on which power transistor is used, in this case the Darlingtone NPN Tip 120 transistor is used and the values were taken from the datasheet. An extra diode is added before the base resistor for safety to the arduino.

3.1.2.1 Circuit

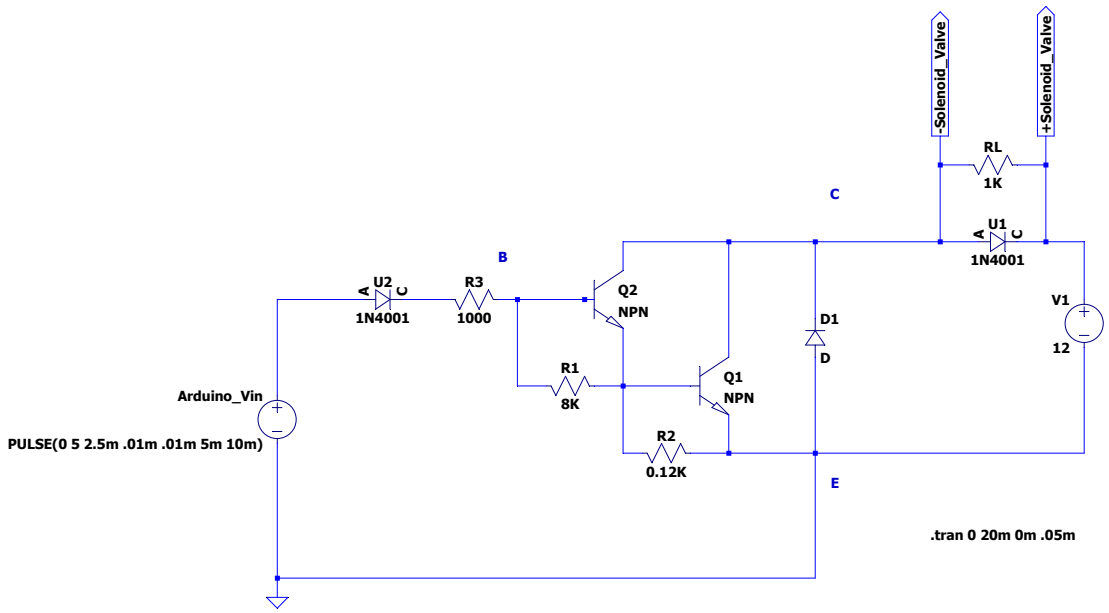


Figure 3.6: Switch Transistor

3.1.2.2 Circuit Simulation

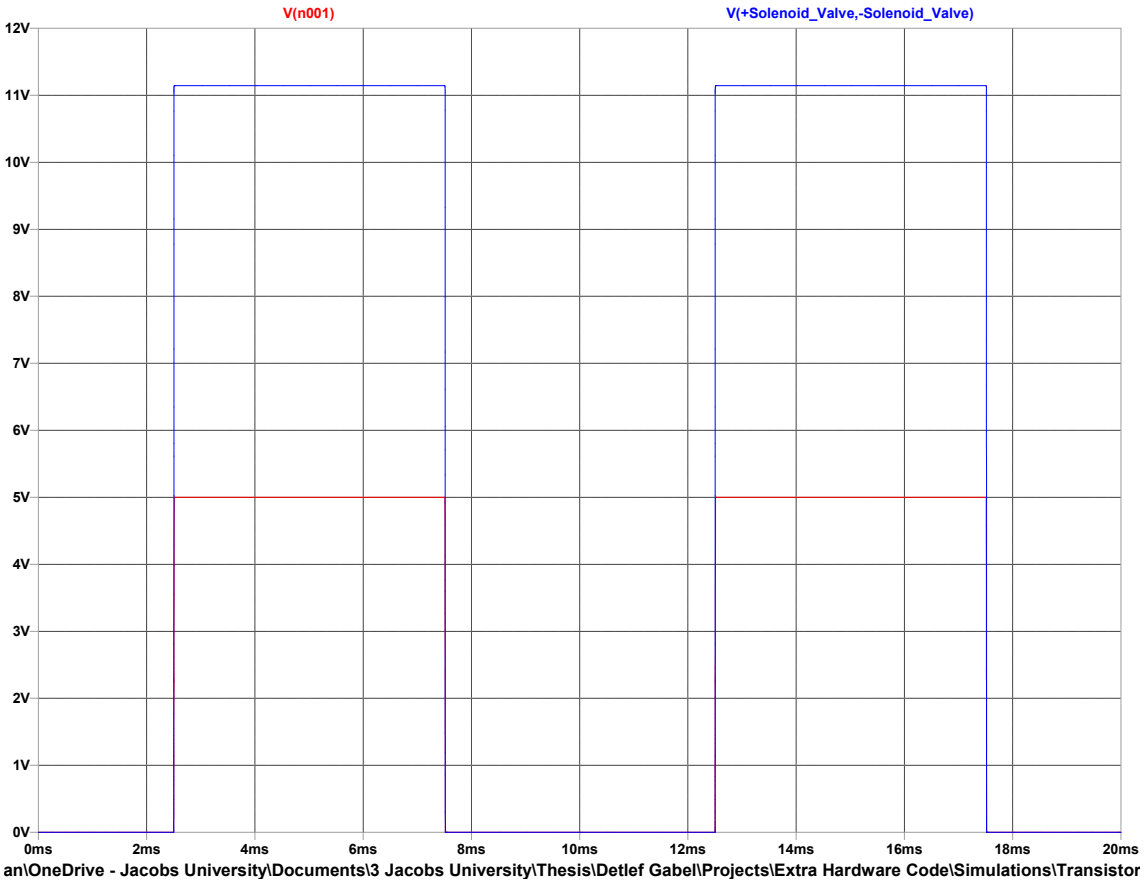
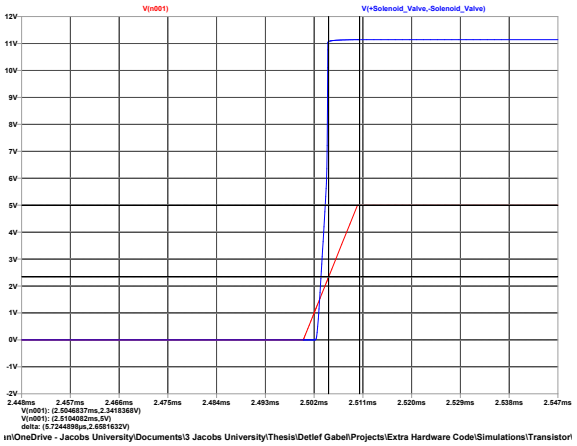
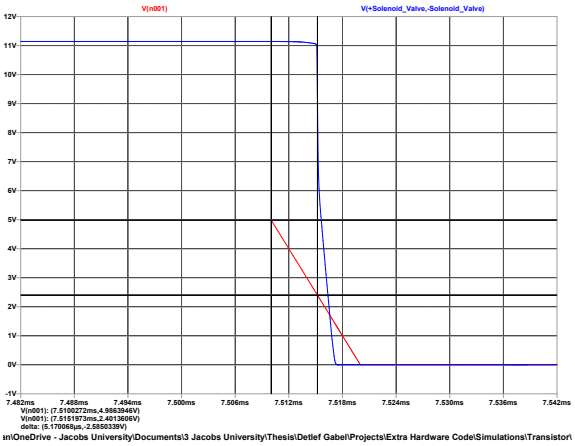


Figure 3.7: Switch Power Transistor Graph, $x\left(t\right)$ = Red, $y\left(t\right)$ = Blue



(a) $t_{on_{risedelay}} = 5.7\mu s$; $t_{rise} = 2.2\mu s$

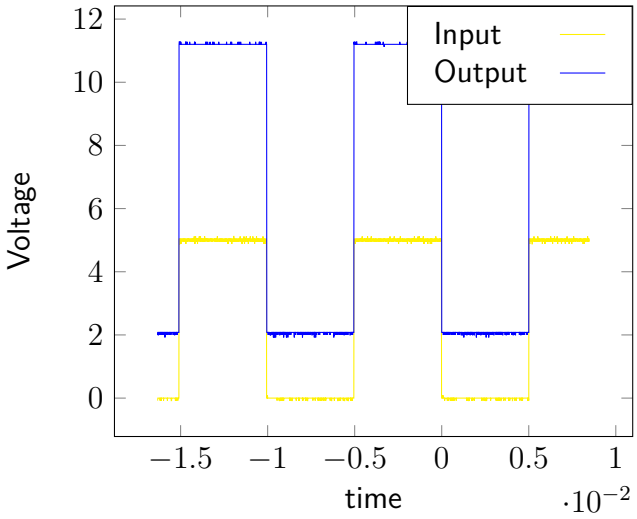


(b) $t_{on_{falldelay}} = 5.17\mu s$; $t_{fall} = 2.5\mu s$

3.1.2.3 Oscilloscope

The circuit was tested. As it can be seen on figure 3.9a and on 3.10 the overall behavior of the output signal is similar to the input signal except for the amplification and the $2V$ DC-offset. This offset doesn't affect the state of the solenoid valve because the threshold is on $6V$. By comparing tables 3.9b and 3.1 the rise and fall properties of the output signal with the switch transistor are tighter to the input signal than the properties of the Op-amp circuit.

This circuit is better compared to the Op-amp with the two power supplies because its delays are smaller, it does not warm up with time causing the current not to drop, not heat sink needed (reducing the thickness of the overall circuit), only one power supply is needed, it is cheaper, faster and the solenoid valve always works with this circuit through the needed time. Those reasons make this circuit design better than the first one.

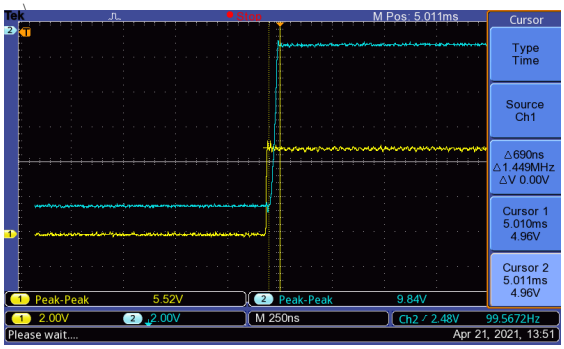


(a) Overall view

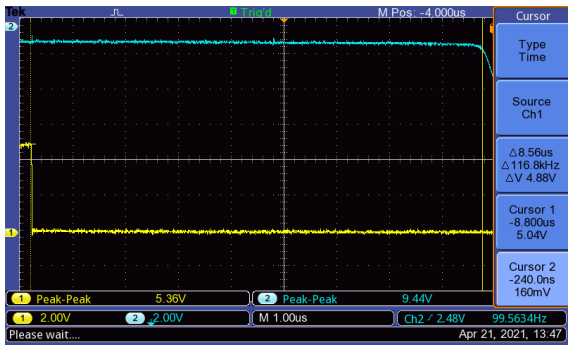
Delay Type	Simulation	Experimental
$t_{on_{falldelay}}$	$5.17\mu s$	$8.56\mu s$
t_{fall}	$2.5\mu s$	$10\mu s$
$t_{on_{risedelay}}$	$5.7\mu s$	$690ns$
t_{rise}	$2.2\mu s$	$10\mu s$

(b) Comparison of the delays for power transistor circuit

Figure 3.9: Experimental Results



(a) Rise,$t_{on_{risedelay}} = 690ns$



(b) Fall, $t_{on_{falldelay}} = 8.56\mu s$

Figure 3.10: Delays

3.2 Interrupt

As specified in the requirements of the gradient mixer, two external interrupt buttons are needed. One button for emergency case, this button will stop the whole process by turning off the pump wirelessly and pause the solenoid valve. A red LED will turn on indicating that the whole process has been detained. The other button will resume the linear gradient mixer process and turn off the red LED once it is resumed. This will indicate that the Linear Gradient Mixer is running back again.

This emergency stop button should only work when the automated gradient mixer process has started and is active, otherwise if the button is pressed the gradient mixer should ignore the interrupt. To be able to have an immediate stop at any given moment during the process. Interrupt Service Routine (ISR) are used. When an interrupt is given, the microcontroller stops its current tasks and executes the ISR. When the ISR finishes, the microcontroller returns to the task it had paused and continues its normal operations [6]. One way to trigger an external interrupt in the Atmega328 is by either pin INT0 or INT1. Any voltage change on one of these pins and it will trigger the corresponding external interrupt and the corresponding flag will be set to 1. Each interrupt has its own ISR vector and it also depends on which pin is being used, therefore the vector should be specified on the code when coding it. This part is done in codeline 7 on the Pseudocode 1.

The red emergency button is connected to pin 20 which is the interrupt pin INT1. This interrupt button is the one that triggers the ISR when it is pressed. The ATmega2560 will then execute the ISR routine, this routine is designed to turn off the pump wirelessly, turn on the red LED, and then it is stuck reading forever the state of the green button until it detects that the green button changes its state. After detecting it, it will turn on the pump again, turn off the red LED and then the Atmega goes back exactly to the task it was running.

3.2.1 Circuit

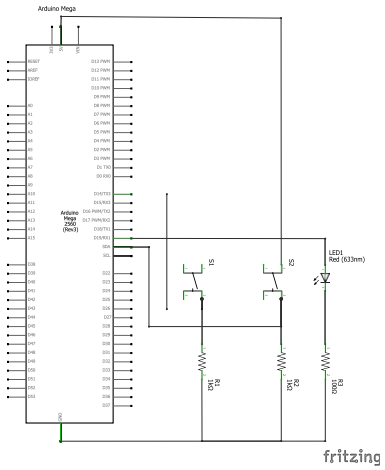


Figure 3.11: Emergency and play buttons sketch

3.2.2 Pseudo-Code

Algorithm 1 Pseudocode for the Emergency Stop and Resume button

```
0: procedure SETUP( )
0:     ...
0:     RedLedPIN = 19
0:     StopButtonPIN = 21
0:     PlayButtonPIN = 14
0:     pinMode(RedLedPIN, Output)
0:     pinMode(StopButtonPIN, Output)
0:     pinMode(PlayButtonPIN, Output)
0:     attachInterrupt(digitalPinToInterrupt(StopButton), PauseProgram, RISING)
0:     ...
0: procedure PAUSE PROGRAM( )
   if Linear Gradient Mixer == Active then
       decider = 1
       PumpAntenna ← Off
       RedLed ← Online
     while decider == 1 do
         buttonstate = digitalRead(PlayButtonPIN)
       if buttonstate == pressed then
           decider = 0
       end if
     end while
       PumpAntenna ← On
       RedLed ← LOW
 end if
=0
```

3.3 Wireless Pump Control

The microcontroller should be able to wirelessly control the pump. The pump is connected to a 433Mhz controlled outlet. To be able to control this outlet with the microcontroller, a 433Mhz antenna transmitter and receiver is needed. First the receiver is connected to the microcontroller, and the Receiver_Demo_Advanced code from the library rc-switch is executed. This code shows on the serial monitor all the signals the 433Mhz antenna is receiving. It shows all the binary code the signal was transmitting and it also gives some properties of the signal as the pulse length, and which protocol is using. By being able to collect all the data being transmitted at that frequency, the signal used to control the pump outlet can be replicated.

First the ATmega2560 should be connected to a 433Mhz antenna and to a computer. It should be running the Receiver_Demo_Advanced code from the rc-switch library. While this is running try to control the pump outlet with the included remote. The Atmega2560 will intercept the message and show the signal properties. These values are noted and then used at the transmitter to emit the exact same signal so it can control the pump.

Recollected data from the intercepted signal: Pulse Length: 718; Protocol: 2; Repeat Transmit: 2; On Code Message = "1000111000001000010111100000000"; Off Code Message = "1000000100001000010111100000000"

3.3.1 Circuit

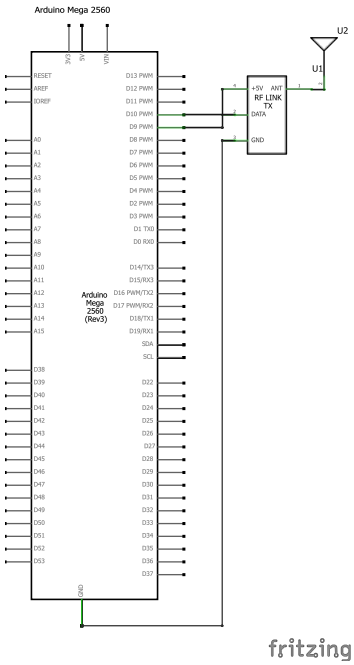


Figure 3.12: 433Mhz Antenna

3.3.2 Pseudo-Code

Algorithm 2 Pseudocode for the Antenna Transmitter

```
0: #include <RCSwitch.h>
0: procedure SETUP( )
0:     .
0:     Serial.begin(9600)
0:     AntennaPin = 10
0:     VCC = 9
0:     pinMode(VCC, Output)
0:     digitalWrite(VCC, High)
0:     mySwitch.enableTransmit(AntennaPin)
0:     mySwitch.setPulseLength(718)
0:     mySwitch.setProtocol(2)
0:     mySwitch.setRepeatTransmit(2)
0:     .
0: procedure PROGRAM( )
0:     mySwitch.send("On Code")
0:     mySwitch.send("Off Code")
  =0
```

3.4 3/2 Way Solenoid Valve

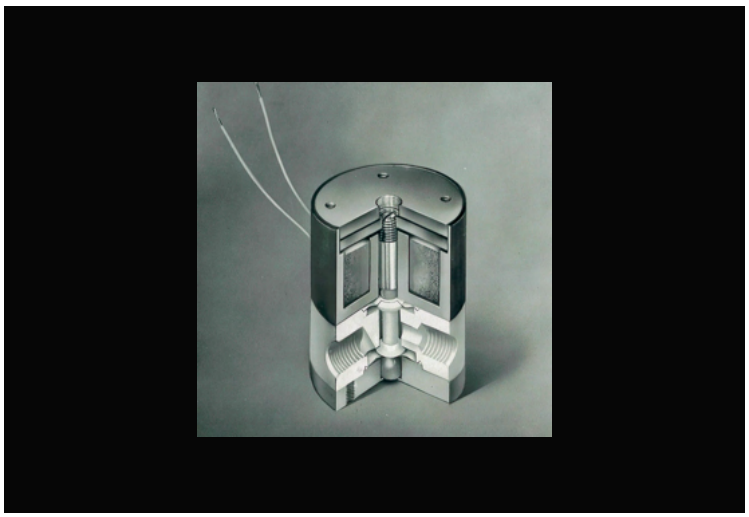


Figure 3.13: 3/2-way NC 12V Solenoid Valve

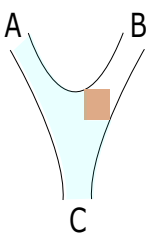
A $12V$ 3/2-way normally closed (NC) solenoid valve is used to control which fluid is flowing. The valve has three ports, A (the non-polar chemical is connected to this port), B (polar chemical), and C (connected to the suction pump), the fluid can only flow either between A and C, or between B and C. The off state of the valve is when there is no voltage or current flow. The fluid flow in this state is between port A and C (non-polar

chemical to suction pump). The on state is when a 12V is applied to the valve, the fluid flow in this state is between port B and C (polar chemical to pump).

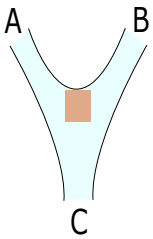
3.4.1 Delay

According to the data sheet of the solenoid valve used, this valve has a delay response time every time it change its state. The response time for changing from off to on is $20ms$, and the response time for changing from on to off is $30ms$.

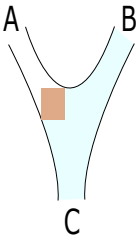
During that delay time, the valve is changing its state as shown in figure 3.14. The flow rate of the initial chemical that was flowing, linearly starts decreasing until it reaches zero percent; and the flow rate of the other chemical starts linearly increasing until it reaches its max flow rate (the pump flow rate). This behavior can graphically be represented by figure 3.15a.



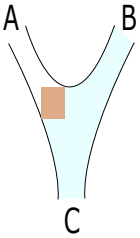
(a) Off: flow rate $= 0$ % polar solvent



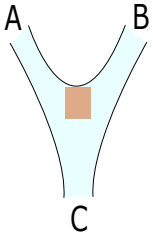
(b) On: during the $20ms$ delay; flow rate $=$ 50% polar solvent



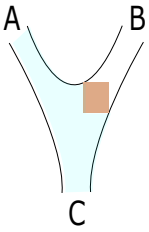
(c) On: after delay, flow rate $= 100\%$ polar solvent



(d) On: flow rate $= 0\%$ non-polar solvent



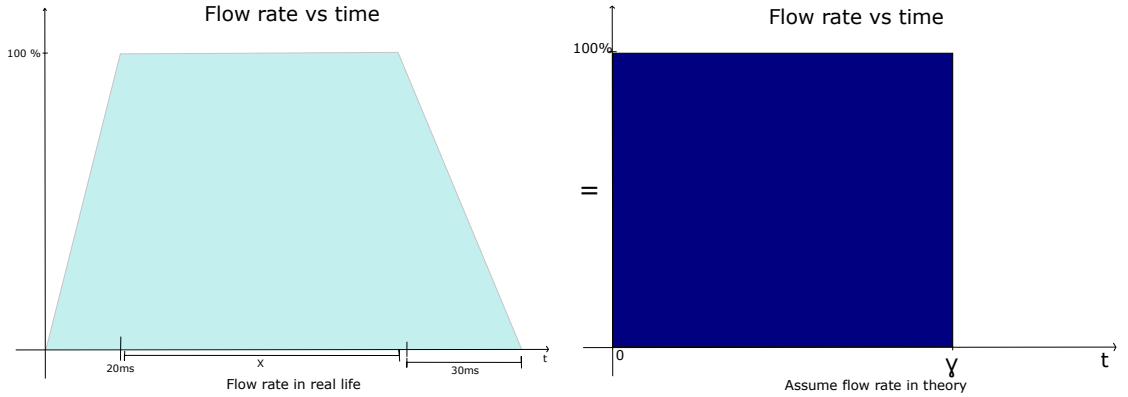
(e) Off: during the $30ms$ delay; flow rate $=$ 50% non-polar solvent



(f) Off: after delay, flow rate $= 100$ % non-polar solvent

Figure 3.14: Visualization of the delay in the change of state. from a-c is from off to on, and from d-f is from on to off

In theory, when the valve changes its state from off to on, then let the polar solvent flow for γ milliseconds and finally change its state again from on to off. The flow rate of the polar solvent is assumed to be constant throughout all the active time as in figure 3.15b. Nevertheless, due to the delays in reality this is not what happens. Therefore, it is needed to find an x millisecond value as the one in 3.15a that represents the exact same amount of the total volume that it will be if it was running for γ milliseconds in the theory graph.



(a) Polar solvent flow rate in real Life **(b)** Theory

Figure 3.15: Flow rate vs time when the valve changes from off to on, runs for a given time, then changes from on to off.

To calculate the x value, which is the time value that the valve should be active after the $20ms$ delay so it can achieve the exact same result as it is expected in theory when it is run by γ milliseconds. The volume area of each graph is calculated, the result are two equations. These equations are equalized and then it is solved for x. These steps are done below.

$$\int_0^{20ms} 5t \cdot dt + \int_{20ms}^{20ms+x} 100 \cdot dt + \int_{20ms+x}^{50ms+x} \frac{10}{3}(-t + x + 50) \cdot dt = \int_0^{\gamma} 100 \cdot dt$$

$$1,000 + 100x + 1,500 = 100\gamma$$

$$x = \gamma - 25ms$$

This means that in reality, to obtain the exact volume of the polar solvent represented in theory by using γ as time, x should be changed from the reality graph to $\gamma - 25ms$. By doing this, it is now possible to calculate the active time λ_{polar}. This is the time that the valve should be activated so it can actually represent the wanted theory γ active time. Since the flow rate between the time $[20ms + x, 20ms + x + 30ms]$ is decreasing, this means that the valve changes its state at the time $x + 20ms$. Therefore, the active time that the $12V$ signal has to last for the polar solvent should be λ_{polar} where:

$$\lambda_{polar} = 20ms + x$$

$$\lambda_{polar} = 20ms + \gamma - 25ms = \gamma - 5ms$$

, The same thing happens for the $\lambda_{non-polar}$, but since in this case the delays are inverted. $\lambda_{non-polar}$ is:

$$\lambda_{non-polar} = 30ms + x$$
$$\lambda_{non-polar} = 30ms + \gamma - 25ms = \gamma + 5ms$$

To be able to maintain both λ positive, the minimum γ value allowed is $5ms$. Resulting in $\lambda_{polar=0}$ or $\lambda_{non-polar=10ms}$. Since, $\lambda_{non-polar=10ms}$ is the only valid value then it becomes the smallest value the valve can be active in a state. It also means that the maximum frequency that this solenoid valve can be used in this experiment is $100Hz$. The minimum active state gives a lower-bound limit in the step by step increment on the linear gradient of 0.83%.

$$\Delta\% = \frac{5ms}{6000ms} = 0.83\%$$
$$f_{max} = \frac{1}{0.010} = 100Hz$$

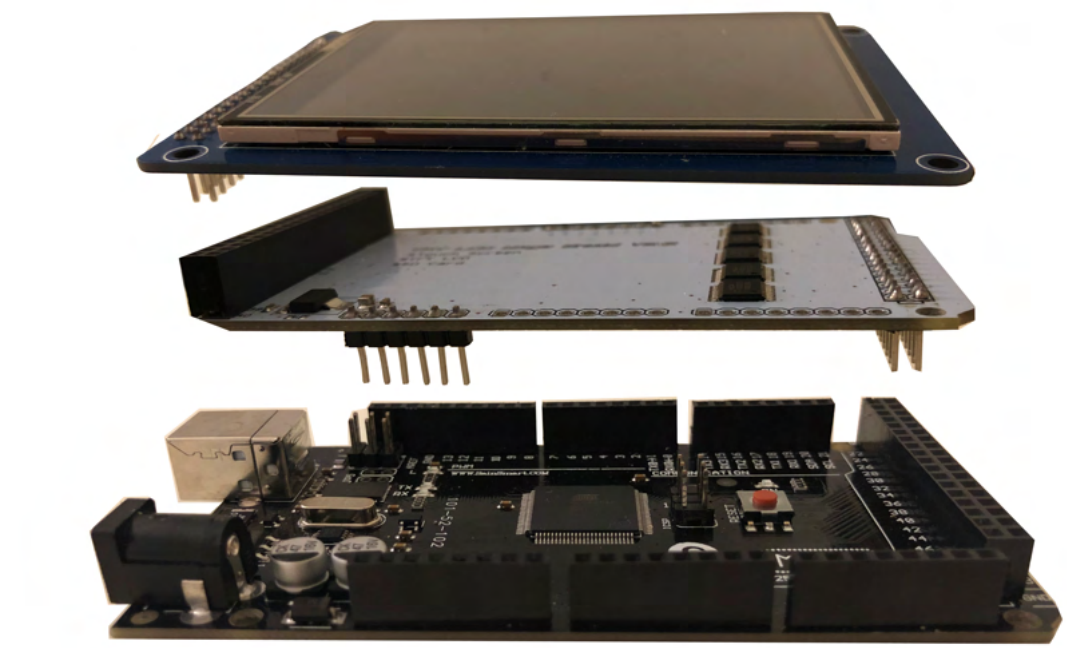
3.4.2 Pseudo-Code

Algorithm 3 Pseudocode solenoid valve

0: **procedure** SETUP( )
0: *pinMode(VALVE, OUTPUT);*
0: *Pump* $\leftarrow$ *off*
0: *VALVE* $\leftarrow$ *off*
0: **procedure** GRADIENT MIXER PROCESS( )
0: $Bandwidth = t_2 - t_1$
0: $n_1 = t_1 \cdot 10$
0: $n_2 = Bandwidth \cdot 10$
0: $delta = (percentage_B - percentage_A)\,/n_2$
0: $time_A = percentage_A \cdot 60$
0: $time_B = 6000 - time_A$
0: *Pump* $\leftarrow$ *on*
0: *VALVE* $\leftarrow on$
0: $delay\,(time_A - 15)$
0: *VALVE* $\leftarrow off$
0: $delay\,(time_B + 5)$
for x := 2; x $< n_1$; x++ **do**
VALVE $\leftarrow on$
$delay\,(time_A - 5)$
VALVE $\leftarrow off$
$delay\,(time_B + 5)$
end for
$time_A = 0$
$time_B = 0$
for x := 1; x $< n_2$; x++ **do**
$intantpercentage_A = percentage_A + delta \cdot x$
$time_A = \text{intantpercentage}_A \cdot 60$
$time_B = 6000 - time_A$;
VALVE $\leftarrow on$
$delay\,(time_A - 5)$
VALVE $\leftarrow off$
$delay\,(time_B + 5)$
end for
Pump $\leftarrow$ *off*
. =0

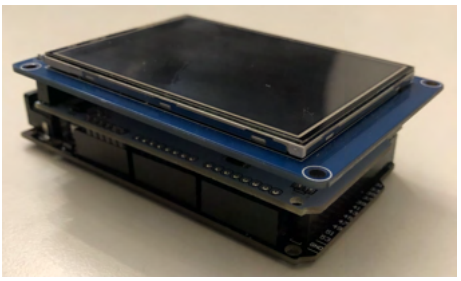
3.5 3.2" TFT LCD Touch screen

To be able to use a 3.2" TFT LCD Touch Screen, a V2.2. shield is needed. This shield, adapts the LCD Touch Screen entrance to the arduino entrance. The connections look as follows:



(a) Gradient Mixer part

(b) Left view



(c) Right view

Figure 3.16: 3.2" TFT LCD Touch screen connected to the Arduino Mega 2560

3.6 Final Circuit

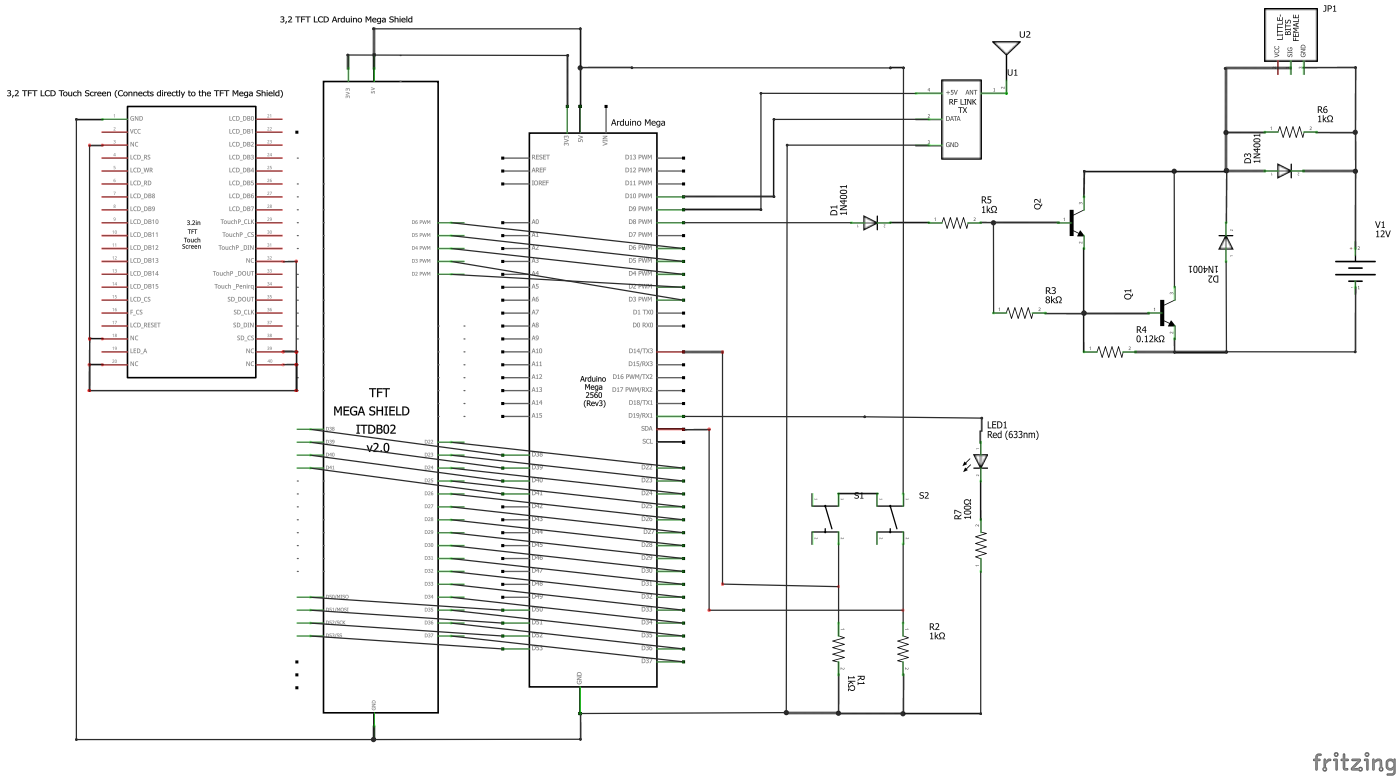


Figure 3.17: Final Circuit, R3, R4, D2, Q1 and Q2 will be replace with a NPN Tip 120 Darlington Resistor in the final PCB (which has the same structure, see data sheet for more information)

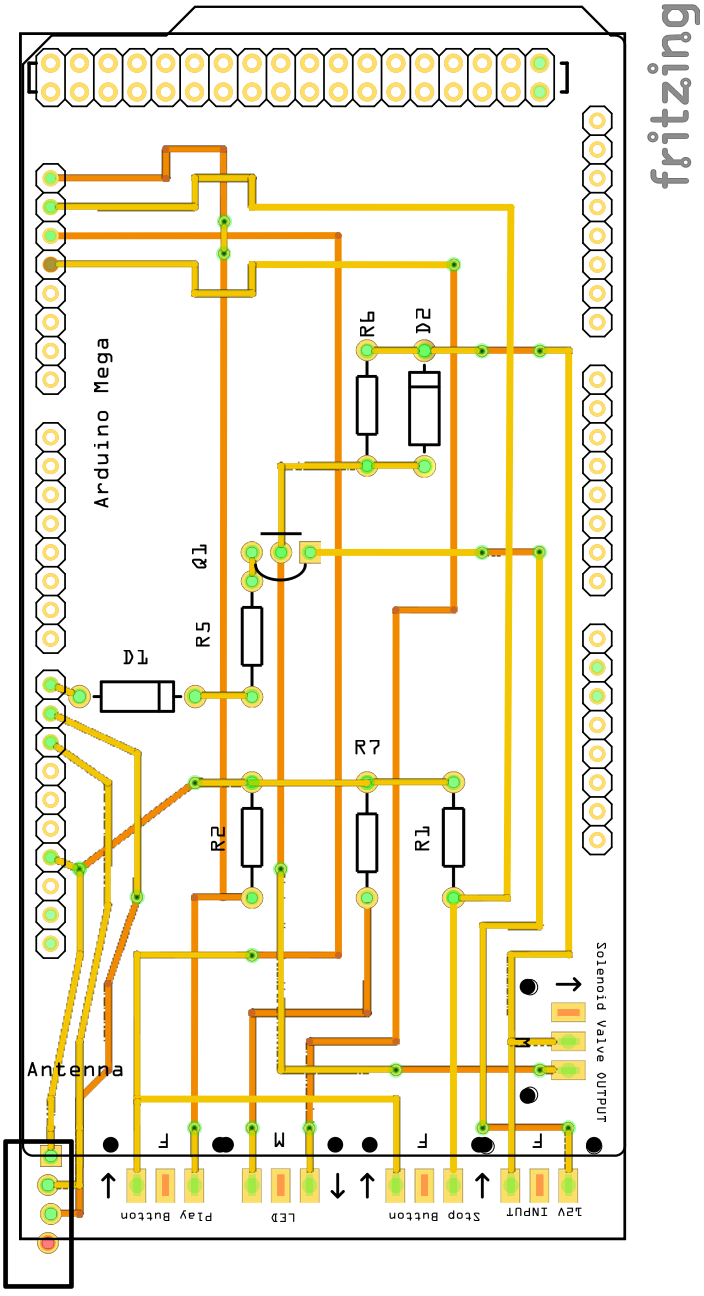


Figure 3.18: Final PCB Circuit, replacing both transistor with one NPN TIP 120 Darlington Transistor

Figure 3.19: Final Product

Chapter 4

Results

4.1 Experiments

In order to see the difference when the delay is compensated to when it is not, an experiment is run for both cases, compensating the delay and not compensating it. In the first experiment the gradient mixer does not compensate the delay. In the second experiment the gradient mixer compensates the delay by using the previous explained steps. Since the percentage changes every 6 seconds, an experiment that can collect samples every 6 seconds and is able to determine the percentage of each solvent in that sample is designed.

4.1.1 Experiment Set-up

A beaker is filled with a polar solvent (methanol). A tube that allows the flow from the polar solvent to port B of the solenoid valve is used. A second beaker is filled with the non-polar solvent (DCM), one side of a second tube is placed inside the second beaker and the other side is connected to port A of the solenoid valve. The third and last port of the solenoid valve, port C, is connected to the pump. The pump sucks the output liquid of the valve and sends it to a fraction collector. The fraction collector was set so it can collect samples every 6 seconds.

To determine the percentage of each solvent, from the samples collected, every two samples were further analyzed by a proton nuclear magnetic resonance. This analysis can quantitatively analyze mixtures containing known compounds [7]. In this case, the 1H NMR identifies the molecules in the solution and it gives the number of hydrogens in each molecule. The solvents used in both experiments are Methanol and DCM, it is known that in a 1H proton NMR spectroscopy methanol shows a peak around 3.46 ppm, while DCM has a peak around 3.97 ppm. This means that the peak to the right will be the methanol peak, and the peak to the left is the DCM peak. The higher the integration value of each peak in the NMR, the more concentration of that specific solvent is in the final solution. The percentage of each solute will further be obtained by integrating the peaks shown on the phase of the proton nuclear magnetic resonance result. This integration value of the peak represents the number of hydrogen atoms in the molecules

of the sample, and by comparing with the theory the percentage can be calculated.

The parameters inserted to the linear gradient mixer are: $t_1 = 0min, A\% = 40\%, t_2 = 4min, B\% = 60\%$. With those parameters, it is known that the linear gradient will take 4 minutes. Since the fraction collector collects samples every 6 seconds, 40 test tubes are needed. After connecting the linear gradient mixer machine to the solenoid valve, the automated process is started.

After letting the gradient mixer do its work, every second sample from the 40 test tubes was collected and sent to the NMR machine for analysis. This procedure was done two times, the first time with the gradient mixer machine not considering the delay of the valve and the second one considering the delay.

4.1.2 Phase of the Proton NMR Spectroscopy

Since the two solvents used are dichloromethane CH_2CL_2 and methanol CH_3OH. It can be observed that dichloromethane has 2 atoms of hydrogen, methanol has 3 atoms of hydrogen plus 1 molecule of hydroxide OH (which has one atom of hydrogen). In the NMR results, three peaks appeared. The integration of each peak signal represents the amount of hydrogen molecule in the substance. In figure 4.1 it is shown the proton nmr result of the sample 22 collected at time 2:12 in the first experiment. As it can be seen, the peak at the right is the methanol peak and it has the 3 molecules, while the peak at the left is the DCM peak, the integration value indicating the number of hydrogen atoms is 1.012. To calculate the DCM percentage in the solution the following formula is used:

$$\% = \frac{Methanol}{3} \cdot \frac{DCM}{2}$$

$$DCM\% = \frac{3}{3} \cdot \frac{1.012}{2} = 50.6\%$$

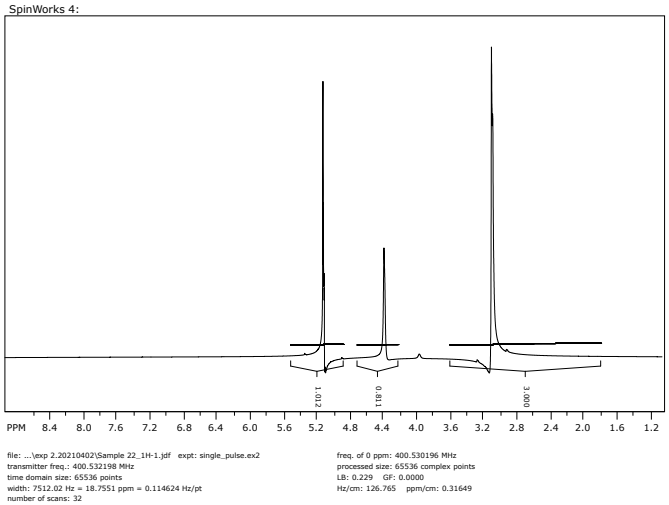
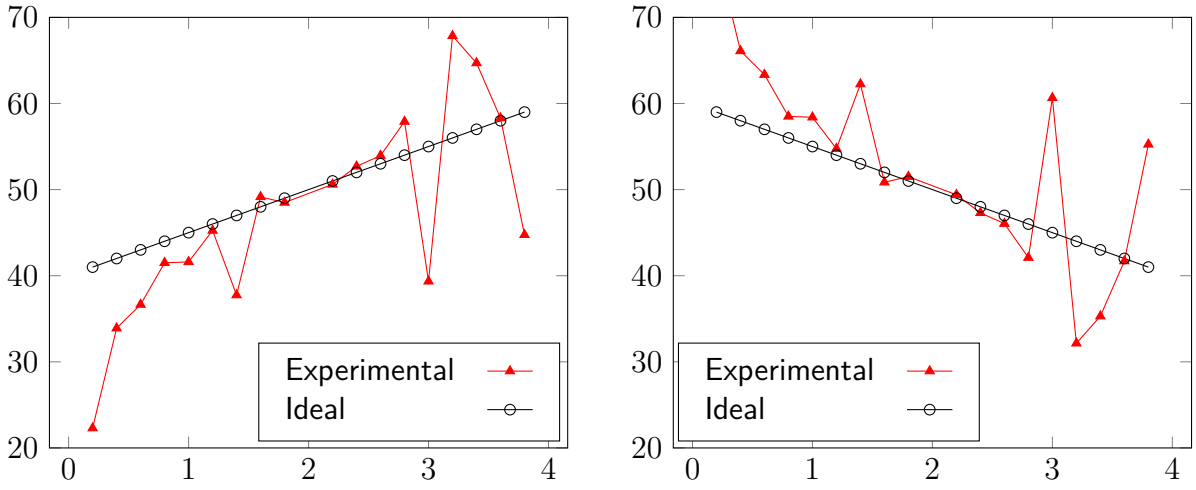


Figure 4.1: 1H Proton NMR result of sample 22 from experiment 1

4.1.3 Graph Results

4.1.3.1 First linear Gradient Mixer experiment

By using the steps explained on section 4.1.1 the first experiment was done. A proton NMR analysis was done to each sample collected and its percentage was obtained as explained before. Unfortunately, the samples 20, and 40 were contaminated while conducting the experiment. In figure 4.2 the graphical representation without the contaminated samples is given.

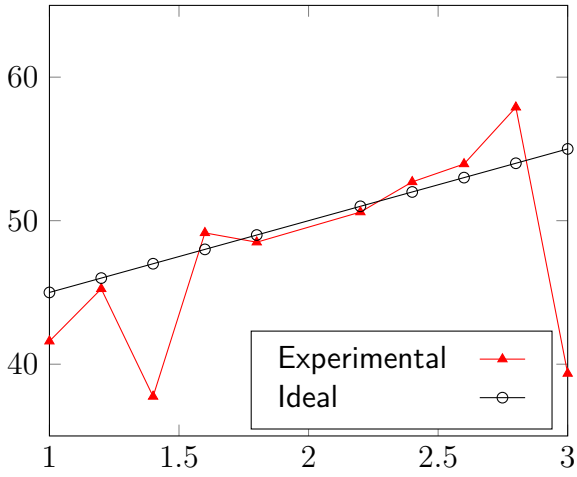


(a) DCM[%] vs time[min]

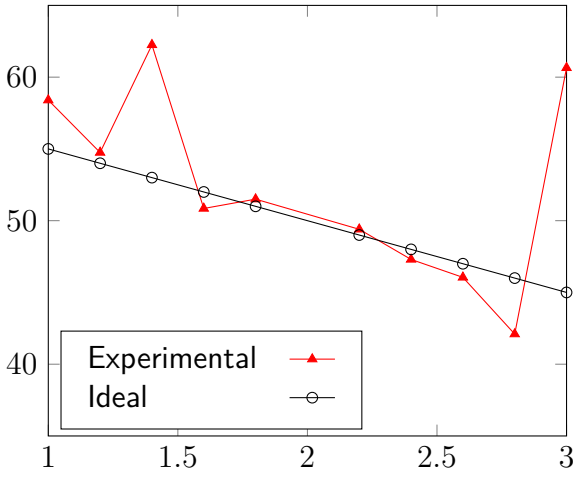
(b) Methanol[%] vs time[min]

Figure 4.2: 1^{st} Gradient Mixer from $(40-60)\,\%$ not contaminated

It can be observed, the experimental values accuracy is high in the middle (between minute 1 and 3) but the repeatability is low. In the first minute, and in the last minute the accuracy and the repeatability are low. This is because in the first minute, as soon the gradient mixer starts, the tubes are empty and filled with air. Once this air is removed after the first minute of fluid flow, the gradient mixer accuracy increases. The air changes the fluid flow therefore it is better to let the gradient mixer flow the first minute and ignore the solvent output from the first minute. The same happens in the last minute. When the gradient mixer is done there is still solvent on the tubes that needs to be delivered to the fraction collector so the pump is still on, but now there is air entering the pipes from the polar and non-polar entrance. Therefore, it is better to ignore the results from the first and last minute. In figure 4.3 the graphical representation when the first and last minute are ignored is shown.



(a) DCM[%] vs time[min]

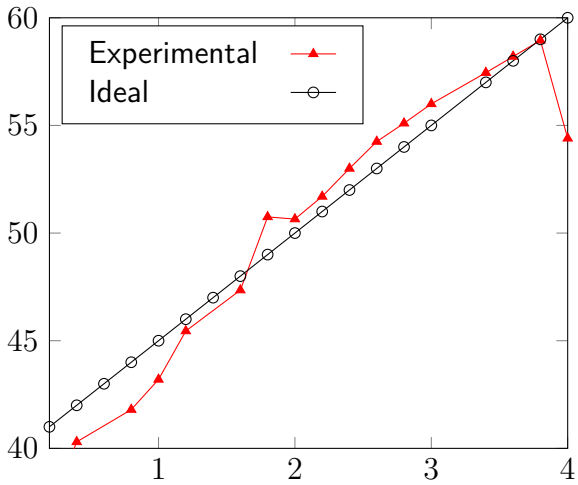


(b) Methanol[%] vs time[min]

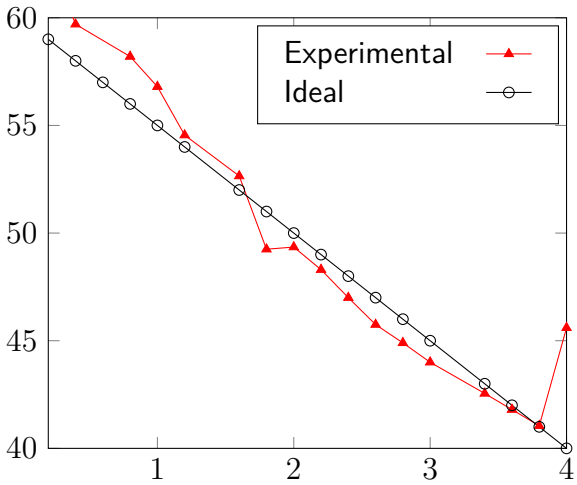
Figure 4.3: 1^{st} Gradient Mixer from $(45 - 55)\,\%$

4.1.3.2 Second Gradient Mixer Experiment

The second experiment was done following the steps from section 4.1.1, but this experiment takes in consideration the delay. Hopefully by taking in consideration the delay the repeatability will improve. The percentages were taken by following the steps from section 4.1.2. On figure 4.4 is shown the results of the second linear gradient mixer.



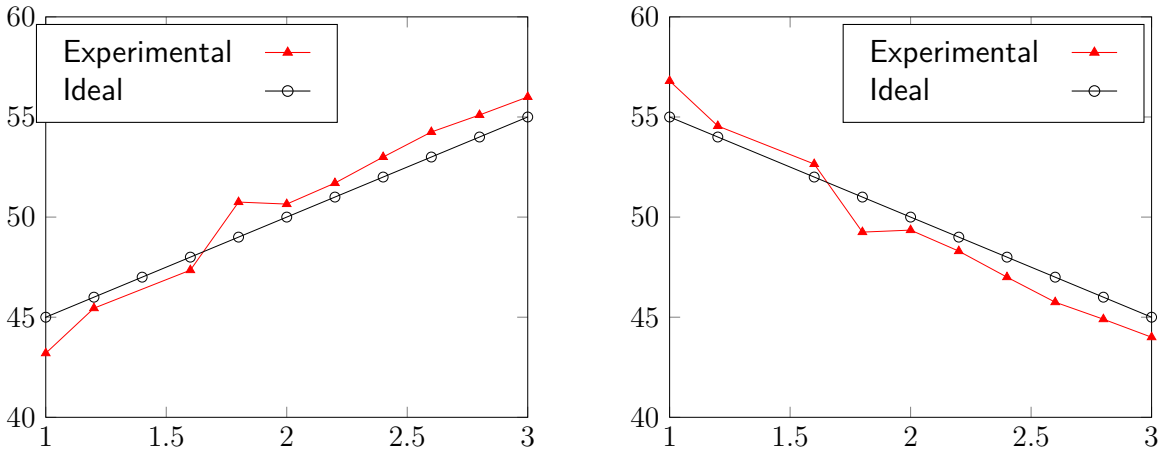
(a) Methanol [%] vs time [min]



(b) DCM [%] vs time [min]

Figure 4.4: 2^{nd} Gradient Mixer run from $(40 - 60)\,\%$ in $4min$

As it can be seen, compare to the 1^{st} linear gradient mixer, now the repeatability is higher. This is because the delay was taken into consideration. Also, due to the same reason as explained on the first gradient mixer experiment the data from the first and last minute are ignore because during these minutes there was air on the tubes (alterating the data). Figure 4.5 shows the linear gradient mixer graph that compensates the delay, and ignores the first and last minute of the process.



(a) Methanol [%] vs time [min]

(b) DCM [%] vs time [min]

Figure 4.5: 2^{nd} Gradient Mixer experiment from $(45 - 55)\,\%$ in $3min$

4.1.3.3 Compare

On figure 4.6, both gradient mixer and the theory graphs are compared. As it can be seen, both linear gradient mixers graph are not tightly bound to the theory graph, this is because of two reasons. The first one, is that in this experiment the step's increment was too high due to the short time (only 3 minutes), resulting in higher changes in the solenoid active time. The gradient mixer was designed to run for longer time (6 hours) so it can have a smaller step increment resulting in a smoother percentage increase. The second one, the NMR machine used also has an instrumental error. Nevertheless, it can be shown that the second gradient mixer (the one taking in consideration the delay) behaves better, by showing better accuracy and repeatability than the first gradient mixer. By taking in consideration the two reasons explained previously, it is expected that the experimental graphs are not tightly bounded to the theory. Although, the final result is close enough to the theory, it can be conclude that the second linear gradient mixer works as expected and it can be used for the task it was designed for.

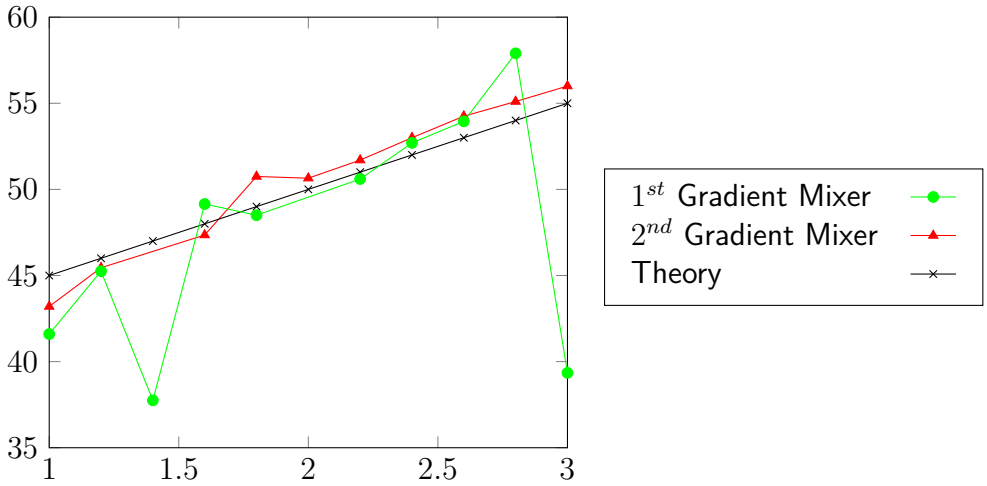


Figure 4.6: Chemical Increase [%] vs Time [min]

4.2 Software

4.2.1 Backend and frontend

The software of the gradient mixer was coded on the Arduino IDE. To be able to control everything on the LCD Touchscreen a different approach of coding is implemented. To be able to interact with the touch screen at any moment. The arduino mega should check if there is any new input at all moment. Therefore, an if statement is the first code line, this if statement will check the state of the LCD touch screen every second. When a touch is detected, the coordinates of the touch are requested and saved on an x and y variable. After saving the x and y coordinates of the touch on the screen, the arduino checks on which page it is on (it can be the homepage, solenoid valve page, process page, etc...) and check which button belongs to the x and y coordinates and execute the command that the button represents. A pseudocode following these steps is given on pseudocode 4.

The homepage contains three buttons, one button to manually control the solenoid valve state, the other one for manually control the pump and the third and last button is used to start the automate linear gradient mixer. This can be seen on figure 4.7. After pressing the automatic button, the user is redirected to the start page. The purpose of this page is to insert the needed parameters for the linear gradient graph, figure 4.9a shows the start page. When this is done, the gradient mixer gives a preview of the graph (figure 4.9b) it is going to follow, if the user accepts this graph then the process starts.

In the process page, everything that is happening is shown. The state of the valve is represented by a yellow box. If the valve is on the polar solvent, then the polar box turns yellow. If it is in the non-polar solvent, the non-polar box turns yellow. Inside this boxes a numerical value is shown, this value means the time the valve will be on that state in milliseconds.

Below the boxes, there is the message section. In this section the gradient mixer prints the message with the progress of job. The first line shows in which cycle from the first stage it is on. These are the amounts of cycles needed on the first stage $(0,t_1)$. It can be seen on figures 4.9c and 4.9d. After finishing the first stage, the line will be erased and a new message is printed saying "Done!", and the second stage starts. The second state is where the linear increase starts, the number of cycle of the state is shown on the next line as shown on figures 4.9e and 4.9f. When this finishes the gradient mixer turns everything off and prints a message indicating that all is done as shown on figure 4.9g.

The design for the manual control pages that controls the solenoid and the pump are shown in figure 4.8.

4.2.1.1 Pseudocode

Algorithm 4 Gradient Mixer Pseudocode

```
0: Import all libraries
0: Initialize the LCD, Touch Screen and RF Transmitter
0: Declaration of Global Variables
0: procedure SETUP( )
0:     Antenna_setup_pseudocode
0:     Emergency_Protocole_setup_pseudocode
0:     Solenoid_Valve_setup_pseudocode
0:     draw_main_menu ()
0:     current_page = 'main_menu'
0: procedure LOOP( )
   if LCD.touchscreen.touch == detected then
        x,y = touch.coordinates
      if current_page == 'main_menu' then
        if x,y == startbutton.coordinates then
             draw_start_page ()
             current_page = 'start_page'
        end if
        if x,y == solenoidbutton.coordinates then
             draw_solenoid_button ()
             current_page = 'solenoid_page'
        end if
        if x,y == pumpbutton.coordinates then
             draw_pump_page ()
             current_page = 'pump_page'
        end if
      end if
      if current_page == 'start_page' then
        if x,y == nextbutton.coordinates then
             draw_enter_page ()
             current_page = 'enter_page'
        end if...
      end if
      if current_page == 'solenoid_page' then
        ...
      end if
      if current_page == 'pump_page' then
        ...
      end if
      if current_page == 'enter_page' then
          draw_graph()
          ...
      end if
      if current_page == 'process_page' then
          solenoid_valve_process_pseudocode
      end if
   end if
=0
```

4.2.2 Interface

The final interface presented at the LCD touch screen is shown on figures 4.7, 4.8, and 4.9.

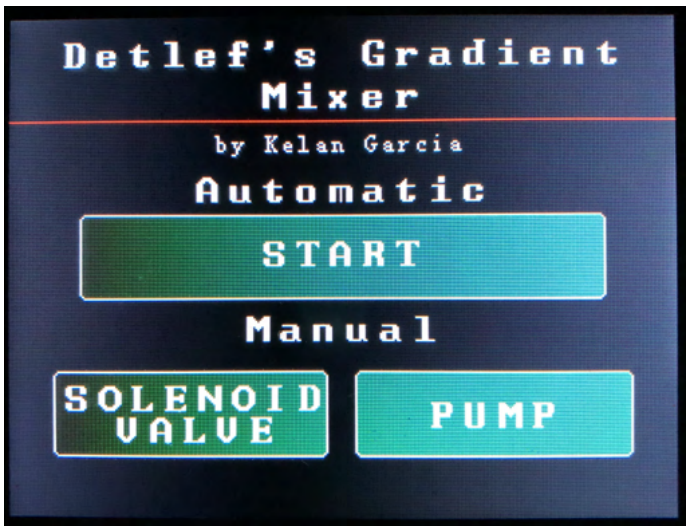
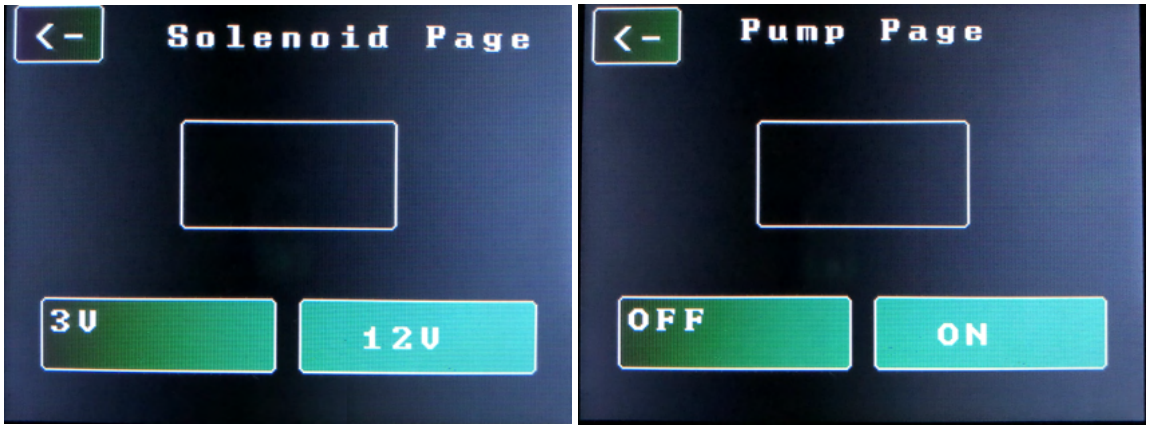
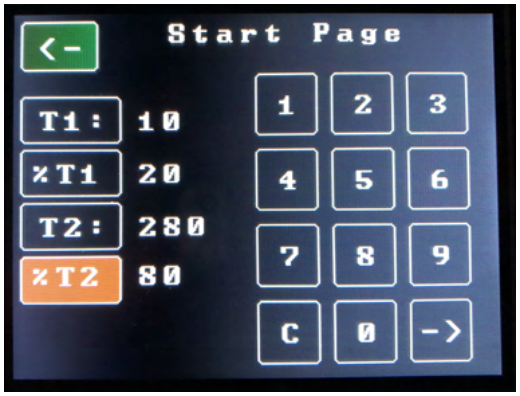


Figure 4.7: Home Page

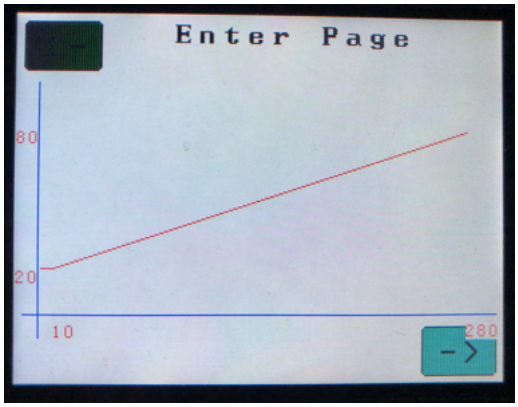


(a) Solenoid Page **(b)** Pump Page

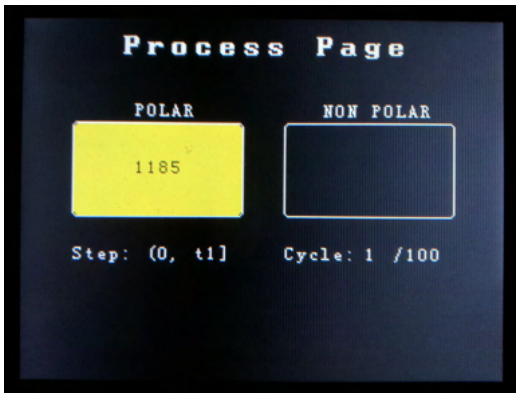
Figure 4.8: Manual Commands



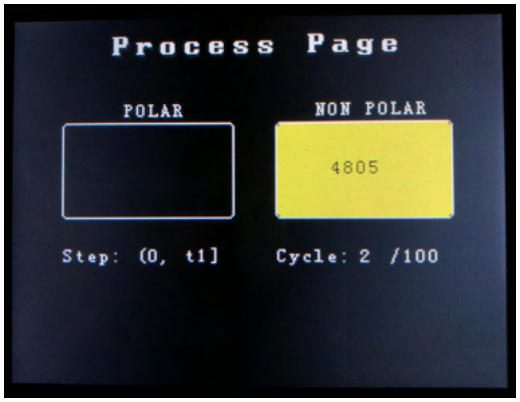
(a) Start Page



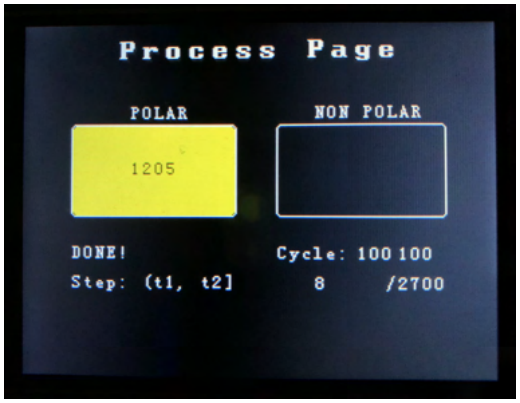
(b) Graph Preview on Enter page



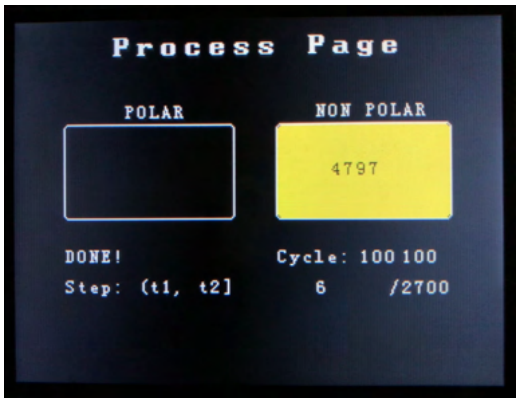
(c) Process Page; Polar solvent flowing and time flow in ms; between 0 and t_1



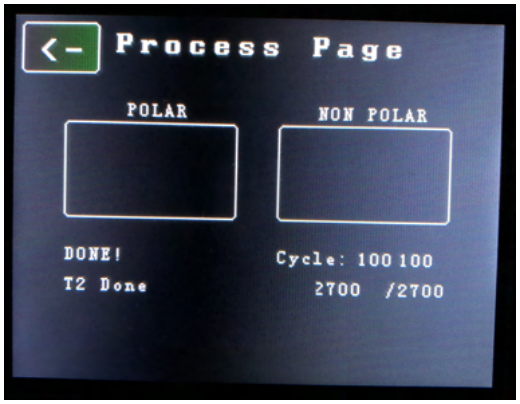
(d) Process Page; non-polar solvent flowing and time flow in ms; between 0 and t_1



(e) Process Page Polar, Polar solvent flowing and time flow in ms; between t_1 and t_2



(f) Process Page; non-polar solvent flowing and time flow in ms; between t_1 and t_2

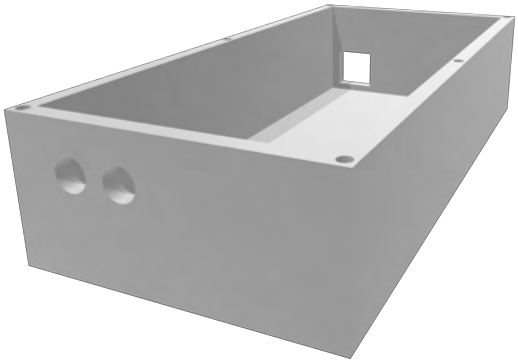


(g) Process Page; t_2 Done, All Process done

Figure 4.9: Automatic control process

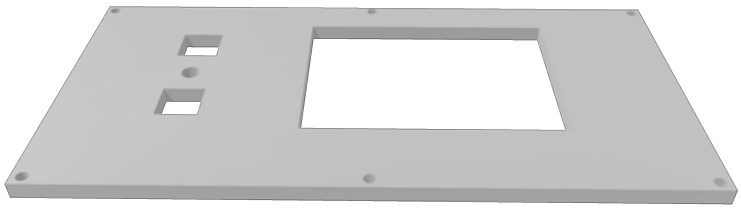
4.3 3D Case

A 3D case is designed to protect and enclose the circuit. The design of this case allows the use of the LCD touch screen, it gives access to the two push buttons, it has a space for the LED, it has a port for the power input, it has a second port for controlling the solenoid valve and it has a last port that gives access to the usb type B of the arduino.



Base

Figure 4.10: Base of the Case



Lid

Figure 4.11: Lid of the Case

Chapter 5

Conclusion

In conclusion, there exist a cheap and reliable way to automate the process of a linear gradient mixer. The main objective of this thesis was to create a machine that can automate the process of creating the solvent needed in the mobile phase on chromatography gradient. This was done with an ATMEGA2560 and a solenoid valve. The first circuit that was design to control the solenoid valve, had some delays and it started to heat up with time causing the solenoid valve to stop changing its state. This was fixed by changing the Op-amp with two transistors creating a switch power transistor. This made a huge improvement to the circuit by making the delays smaller, eliminating the need of a heat sink to cool down the circuit, and by reducing the cost of the whole project since transistors are cheaper than Op-amps. The linear gradient mixer was tested at the end to see if it behaves as expected. The final result was as expected and it was close enough to the theory. To improve the final result, the use of a sensor that can calculate the percentage composition of the polar and non-polar solvent on real time can be investigated. This sensor can be implemented in a close loop form so it can correct the output instant percentage concentration. If this is done, then the BIBO stability of the machine has to be investigated.

Bibliography

[1] L. Ettre and A. Zlatkis, *75 Years of Chromatography: A Historical Dialogue*, ser. ISSN. Elsevier Science, 2011. [Online]. Available: https://books.google.de/books?id=tdrFTHQ3e64C

[2] B. Bickler, "What is a Chromatography Gradient?" 2019. [Online]. Available: https://selekt.biotage.com/blog/what-is-a-chromatography-gradient

[3] C. BCR, "Controlling A Solenoid Valve With Arduino," 2015. [Online]. Available: https://bc-robotics.com/tutorials/controlling-a-solenoid-valve-with-arduino/

[4] T. Intruments, "LM741 Operational Amplifier Datasheet," 2015. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm741.pdf?ts=1620889666221&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLM741%253FHQS%253DTI-null-null-octopart-df-pf-null-wwe

[5] P. Scherz and S. Monk, *Practical Electronics for Inventors*, ser. ISBN. McGraw-Hill Education, 2016.

[6] P. F. Hu, "External Interrupt (Assembler)," 2015. [Online]. Available: http://embsys-fhu.user.jacobs-university.de/?page_id=13

[7] X. Li and K. Hu, *Quantitative NMR Studies of Multiple Compound Mixtures*, 09 2016.