Table of Contents

User Guide

1.0 Introduction

The resulting product of this project is a fluid processing controller that emulates a small-scale industrial process. It involves the acquisition and control of process variables with fluids using basic components for fluid level and temperature measurement such as pressure and temperature sensors. The system includes two liquid storages, a processing tank, and both physical and virtual control panels. Additionally, two water pumps are used to control the level in the processing tank coming from the storage tanks, a heating resistor is used to warm the water to the temperature that the user sets and an electro valve is utilized to serve the liquid when the process is finished. The genesis of this project is my interest in the food manufacturing industry and my desire to eventually have my own food processing business.

2.0 Setup and Operation

To properly configure and operate this product, it is necessary to connect both 12V and 24V power supplies to the PCB, as well as the temperature and pressure sensors and all actuators to their corresponding headers on the PCB. The Raspberry Pi must be powered on and its serial ports connected to the PCB, while also ensuring it has internet connectivity. To begin the process, the user should navigate to the Raspberry Pi's IP address, which can vary depending on the network's DHCP settings, and access the form page at /fluids_project/web_server/form.html. For example, if the Raspberry Pi's IP address is 10.0.0.83, the user should browse to http://10.0.0.83:8080/fluids_project/web_server/form.html. On the form page, the user can enter their desired settings for the process, taking note of the restrictions and limits set for the parameters to optimize the system. Once the user has submitted the form, they can start, stop, and monitor variables such as temperature, processing tank volume, the state of start, stop and serve, as well as the current step of the process being executed. These are the different steps of the system and their meaning:

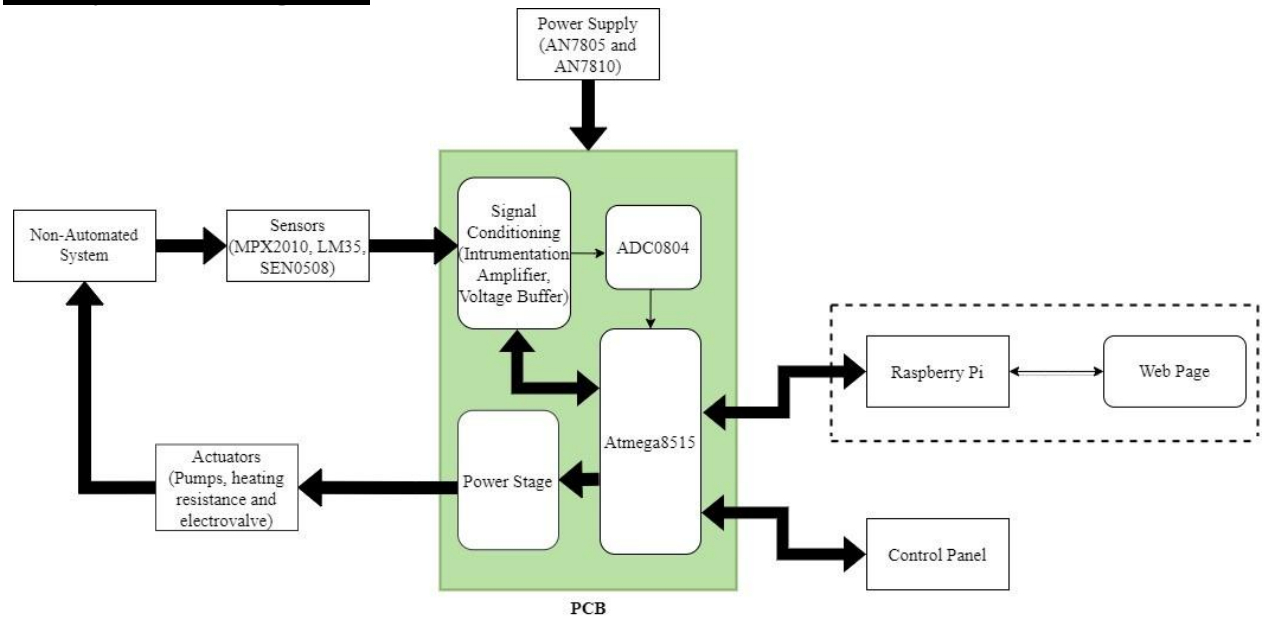
- Step 0: Microcontroller is waiting for serial values from Raspberry Pi.
- Step 1: The values sent from the Raspberry Pi are transferred to process variables.
- Step 2: Water Pump 1 is started.
- Step 3: Water Pump 1 is stopped.
- Step 4: Water Pump 2 is started.
- Step 5: Water Pump 2 is stopped.
- Step 6: Heating resistance is started.
- Step 7: Heating resistance is stopped.
- Step 8: Cooldown counter is executed.
- Step 9: Liquid can be served (electro-valve is opened).
- Step 10: Electro-valve is closed.

3.0 Specifications

Parameter	Minimum	Maximum	Unit
Input Voltage (Control circuit)		5	V
Input Voltage (Power circuit)		24	
Storage Tank Volume		1300	ml
Water Pumps Voltage	6.3	16.8	V
Water Pumps Current	0.27	0.83	I
Water Pumps Power	1.7	14	W
Temperature in Processing Tank	10	60	°C
Heated Liquid Temperature	23	30	
Heating Resistor Voltage	4.65	22.9	V
Heating Resistor Current	0.18	0.82	I
Heating Resistor Power	0.84	18.78	W

Technical Guide

4.0 System Diagram



5.0 System Description

- **Sensor Module:**
    - **Pressure Sensor:** The purpose of the MPX2010 pressure is to provide precise and reliable detection of the liquid level in the processing tank. This sensor works by measuring the pressure difference between its two ports, one exposed to the atmosphere and the other one exposed to the liquid in the tank, which changes the electrical resistance of the sensor as the liquid level changes. The intended operation is to ensure that the liquid level remains within the intended optimal ranges, while enabling accurate monitoring of the processing operation.

    - **Temperature Sensor:** The purpose of the LM35DZ waterproof temperature sensor is to accurately measure and monitor the temperature of the liquid in the processing tank, and convert it into the corresponding electrical signal. The sensor's output is linear, and proportional to the temperature in Celsius, with a scale factor of 10mV/ °C, which allows accurate and reliable temperature monitoring, and ensures that the processing tank operates within the desired temperature range.

    - **Capacitive Sensors:** This module is designed to detect liquid level in the tanks using SEN0506 capacitive sensors. The purpose of this module is to determine whether the liquid level in the storage tanks is high or low, while in the processing tank, it will detect only the high level. The capacitive sensor used in the processing tank is as used as a redundancy method, since it works alongside the pressure sensor

guaranteeing that the high liquid level in the processing tank is always detected and providing an extra layer of protection.

- **Signal Conditioning Module:**
    - **MPX2010 Signal Conditioning:** The purpose of the signal conditioning module is to prepare the voltage signal obtained by the MPX2010 pressure sensor for use with the Analog-to-Digital converter. To achieve this, two LMC6062 OP-AMPs are connected to the sensor to server as an instrumentation amplifier. The instrumentation amplifier allows to filter out any noise present in the signal, adjust the offset, and adjust the gain before the signal is connected to the A/D converter. The intended operation is to ensure that the signal is optimized for the best possible measurement accuracy, and to minimize signal distortion.

    - **LM35 Signal Conditioning:** The signal conditioning module for the LM35DZ temperature sensor prepares the analog signal for conversion into digital form by the ADC. Its purpose is to ensure accurate temperature readings by amplifying the signal, performing calibration, and linearization. By incorporating a gain of 5, this module enhances the signal's sensitivity for better resolution and sensitivity. This module plays an important role to maintain the quality and consistency of the processing tank operation.

- **ADC0804 Module:** The purpose of the ADC0804 module is to convert the amplified analog voltage signal obtained from the instrumentation amplifier into digital format that can be read and processed by the Atmega8515 microcontroller. In addition to the pressure sensor's ADC, another ADC0804 integrated circuit has been added to handle the analogto-digital conversion of the LM35DZ temperature sensor. The purpose of this module is to convert the voltage signal from the LM35DZ temperature sensor, which has already been amplified by the signal conditioning module, and the liquid level in the processing tank into digital data that can be processed by the microcontroller, providing accurate and reliable temperature and liquid level data for monitoring the processing tank operation.
  This module's intended operation is to sample and convert the analog temperature signals and liquid level into digital data when indicated by the program.

- **Power Supply Module:** The power supply module comprises two regulated power supplies. The first is a 12V DC plug power supply adaptor that serves as the input to an LM7810 regulator, which supplies a stable 10V to the instrumentation amplifier. The same 12VDC plug also serves as the input to an LM7805 regulator, which provides a reliable 5V supply for the microcontroller, ADC, LM35 temperature sensor, and its signal conditioning module, a voltage buffer based on an operational amplifier. Additionally, a DC 24V 10A Power Supply Adapter Transformer Regulated Switch is utilized to power the actuators, which include the two water pumps, heating resistance, and electro valve. The power supply module not only provides the necessary voltage to operate all modules but also ensures the separation of the control and power sections of the circuit for enhanced safety. This

separation is important given that the control section operates with 5V while the power section requires 24V.

- **Power Stage Module:** The power stage module is composed of four circuits, each dedicated to one of the four high-power actuators, all powered by 24V. The purpose of this module is to control the actuators' power supply safely and efficiently using lower-power signals from the Atmega8515 microcontroller. To achieve this, optocouplers are used to isolate the high-voltage actuators from the low-voltage microcontroller, preventing voltage spikes and protecting the microcontroller, and allowing safe and efficient control of the power supply. The MOSFETs act as electronic switches that regulate the current flow to the actuators. The operation of the chopper circuit is synchronized with the microcontroller's PWM output signal, which serves as the input to the optocoupler and regulates the gate voltage of the MOSFETs. The chopper circuits use this switching mechanism to efficiently regulate the actuators' power supply while ensuring their safe operation and providing reliable control of the actuators.

- **Actuators Module:**
    - **Water Pumps:** The purpose of the two water pumps is to transfer liquid from the storage tanks to the processing tank through pipelines. This transfer process is accomplished by regulating the speed of the water pumps through the chopper circuit, which is controlled by the Atmega8515 microcontroller's PWM signal. The intended operation of the water pumps is to accurately adjust their speed according to the signal received from the power stage module ensuring a precise liquid transfer.

    - **Heating Resistance:** The heating resistance is responsible for raising the temperature of the processing tank's liquid to a programmed value by converting electrical energy into thermal energy. The Atmega8515 microcontroller sends a PWM signal to the power stage module, which then controls the power delivered to the heating resistance. Overall, the intended operation of the heating resistance is to provide precise control of the liquid temperature through the microcontroller's PWM signal and the power stage module.

    - **Electro valve:** The electro valve serves as a security feature that prevents liquid from being served until the entire process is complete, and the user-defined cooling time has elapsed. The microcontroller controls the electro valve, allowing it to operate in two states: ON or OFF. If the conditions are not met, the electro valve remains OFF. The electro valve is connected to a regular port on the microcontroller since it only has two states, and also passes through the power stage module. Overall, the intended operation of the electro valve is to provide an added layer of safety and control to the processing tank's liquid flow.

- **The Atmega8515 Module:** This module serves multiple purposes in the system. First, it is responsible for processing digital data obtained from the ADC0804 module, more specifically from the pressure and temperature sensors, which are passed through the signal

conditioning and ADC modules. Additionally, This module also controls the actuators with the help of three PWMs which are configured on the microcontroller: two 16-bit PWMs which serve the two water pumps, and one 8-bit PWM that controls the heating resistance. The signals are sent to the power stage module, which then controls the actuators. Another of its responsibilities is the sequential control of the system. The sequential control involves a series of steps that must be followed to complete the processed, as long as all the necessary conditions are met. If any of the conditions in the step is not met, the process will not proceed to the next step. This sequential process is comprised of 11 steps:

- Step 1: The microcontroller waits for the process values from the Raspberry Pi, as well as the start signal.
- Step 2: The serial communication variables are transferred to the process variables. ○ Step 3: Water pump 1 is started with the user-defined value.
- Step 4: Water pump 1 is stopped.
- Step 5: Water pump 2 is started with the user-defined value. ○ Step 6: Water pump 2 is stopped. ○ Step 7: The heating resistance is started with the user-defined value. ○ Step 8: The heating resistance is stopped. ○ Step 9: The cooldown time counter is executed. ○ Step 10: Liquid is served after the serve signal is sent by user (electro-valve is opened). ○ Step 11: Electro-valve is closed after the pressure sensor detects that the water level is low.

The control sequence is dependent on meeting the necessary conditions and status of the start and stop signals. If the user sends a stop signal on the web server or physical control panel, the process will stop until a start signal is sent. The process then restarts from the step where it was stopped. Additionally, the process checks the water level of the storage tanks. If an empty tank is detected, or an error with the storage tank sensors is identified, the process will also stop.

The microcontroller is also responsible for the two-way serial communication with the Raspberry Pi. It reads two bytes sent by the Raspberry Pi to obtain the initial user-defined process values, where the first byte corresponds to a command identifying the type of data being sent, and the second byte is the data value. The same principle is applied when sending data from the microcontroller to the Raspberry Pi. The data values sent include the step, temperature, volume, start signal state, stop signal state, serve signal state and cooldown counter state.

- **Raspberry Pi**: The Raspberry Pi module serves two main purposes, which are hosting the web page, and communicating with the Atmega8515 microcontroller.
  - **Web Page**: The web page is hosted using Apache 2 to provide a web server that allows direct interaction with the user. The main page, called form.html, prompts the user for multiple parameters that define the system's behaviour, including the percentage of liquid from each tank, the speed of the water pumps, the desired temperature, the power percentage for the heating resistors, and the desired cooldown time before serving the liquid. The data from this page is then submitted to the formresults.php file, which comprises HTML, PHP, and JavaScript code. This

page handles all the backend server process that is needed to effectively communicate with the Atmega8515 microcontroller.

  - **Logic Level Converter:** This module facilitates the communication between the Atmega8515 microcontroller and the Raspberry Pi. This is achieved by converting the different voltage signals between the 3.3V logic level of the Raspberry Pi and the 5V logic level of the Atmega8515.

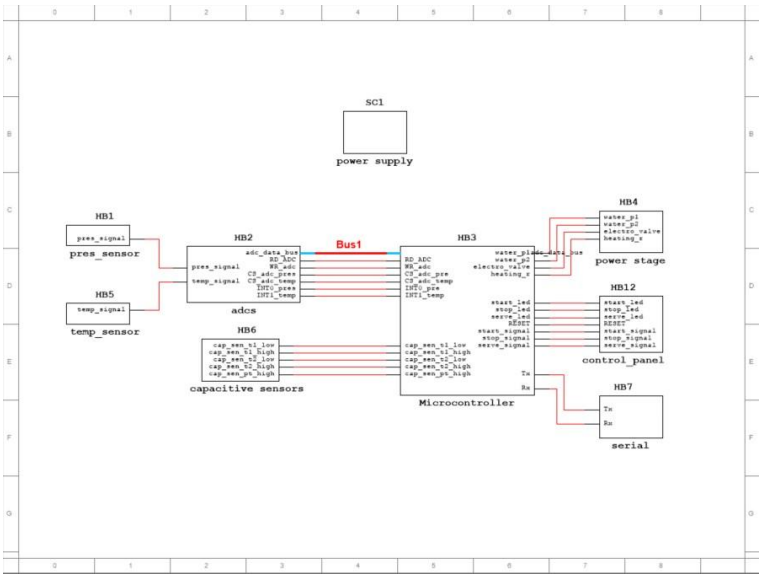
- **Control Panel:** The control panel module serves as a physical redundancy element for the buttons on the web page, enabling control over the system's sequential steps by sending start, stop, and serve signals, just like the web page buttons. The control panel has the same functionality as the web page buttons but it offers added flexibility. In case of a web crash, unavailable internet connection, or failure to send data, the control panel allows for continued control over the system, enabling it to be stopped if necessary, and started without needing to use the web page. Additionally, the control panel includes three LEDs: start, stop, and serve, which indicate the current status of the industrial process.

- **PCB:** The purpose of the PCB (Printed Circuit Board) is to provide a reliable connection for the multiple electronic components, including the microcontroller, ADCs, sensors, MOSFETs, actuators, and all other devices. The main PCB is divided into two parts: the control section on the left, which is comprised of the microcontroller and low-power devices, and the power section on the right, which is composed of the 24V-powered devices that provide control over the actuators. These two sections are separated by optocouplers that provide both electrical and visual isolation. The second PCB is the control panel which comprises the buttons and LEDs, and it is connected to main PCB using headers and wires.

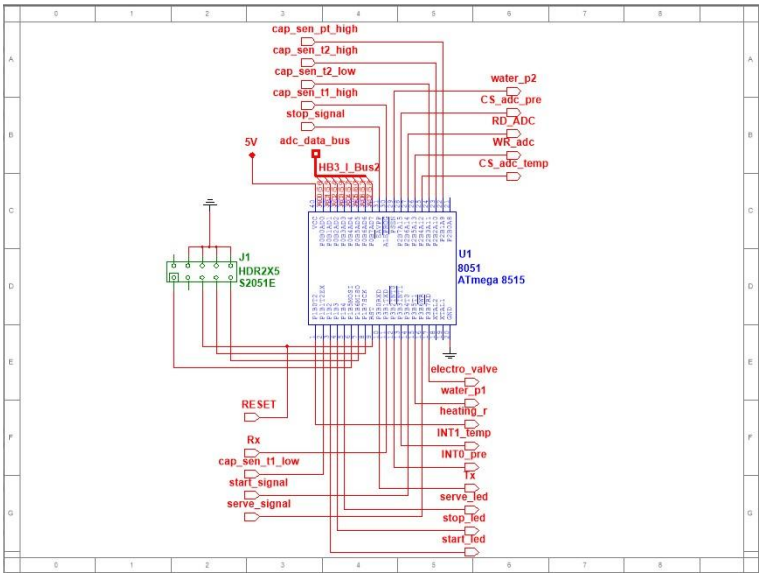
- **Non-automated Module:** This module consists of various components that are part of the system but do not belong to the electrical circuitry such as the structure, the liquid storage tanks, the processing tank, and pipelines. One port of the MPX2010 pressure sensor is connected to a tube that is taped to the bottom of the processing tank to measure the pressure level of the liquid in the tank. ¼ inch vinyl tubing is being used to measure the pressure in the processing tank. The processing tank, along with two 4L storage tanks and pipelines, are also included in this module. The water pumps are connected to the storage tanks and the processing tank via 1/2 inch vinyl tubing, while the processing tank is connected to the electro valve using 3/8 inch vinyl tubing Additionally, manual valves were added to the processing tank to optimize the liquid's flow and to act as redundancy components.

6.0 Circuit Description

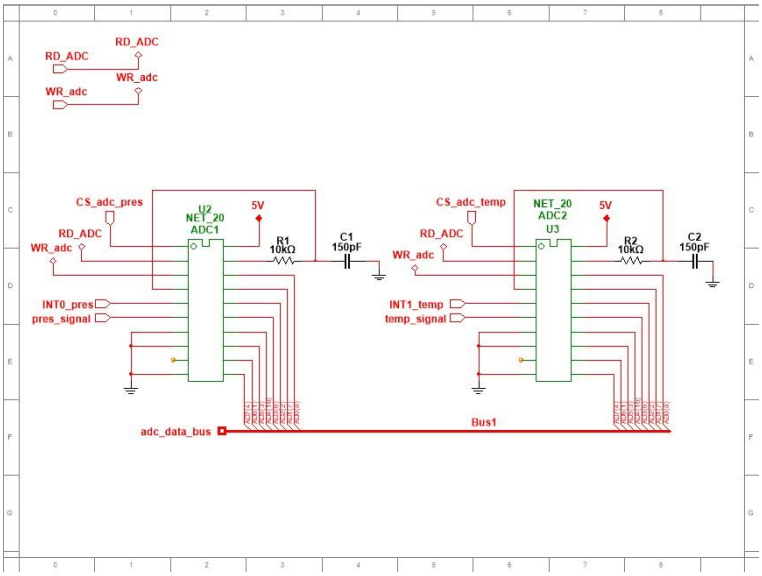
- **System's Circuit**



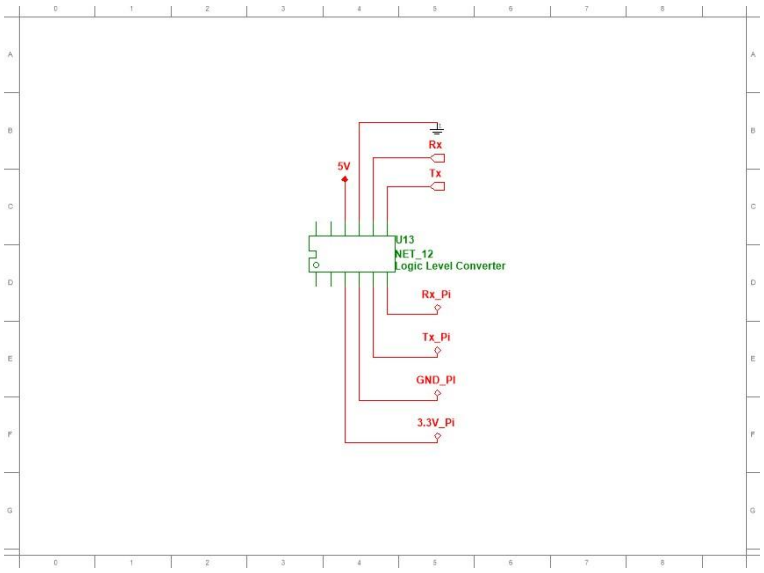
- **Microcontroller:** The control panel is connected to the microcontroller using regular ports configured as input for the start and serve signals, and as output for the LED signals. However, the stop button is connected to INT2 since it is a signal that requires immediate execution in the code and cannot be executed using the polling method. The logic level converter is connected to the Rx and Tx pins on the ATmega8515 which are part of the USART interface. The ADCs are connected to the address port, the control signals are connected to regular ports, and the ADC interrupts are connected to the interrupts on the microcontroller. The actuators are connected to the PWM pins, except for the electro-valve which is using a regular port.



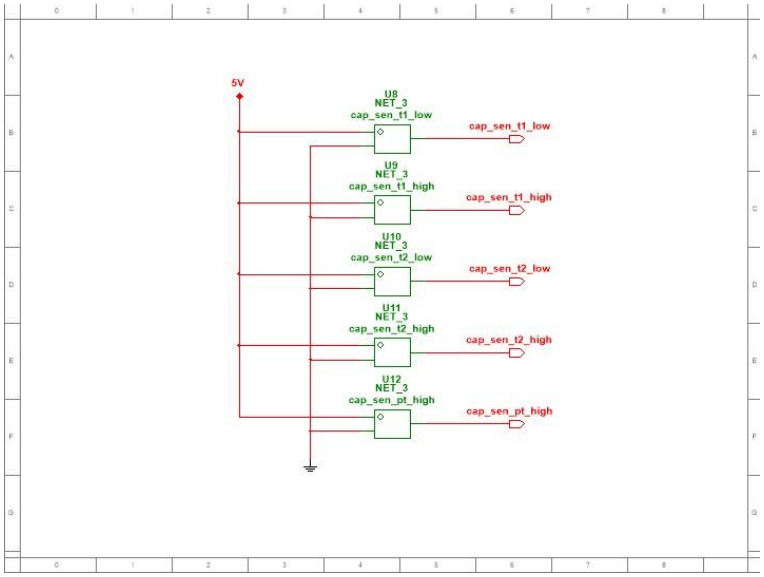
- **ADCs:** The data buses of both ADCs are connected to the same Port A of the Atmega8515 microcontroller, enabling the microcontroller to read data from each ADC. The write (/WR) and read (/RD) signals of both ADCs are also connected to the same /RD and /WR pins of the microcontroller, allowing the microcontroller to have control over their operation. The Chip Select (/CS) signal of each ADC is connected to different ports on the microcontroller, enabling the microcontroller to select which ADC to communicate with and read data from. The two ADCs are configured to run as interrupt-driven, meaning that when an ADC completes a conversion, it generates an interrupt signal on its INT pin which are connected to the microcontroller's INT0 and INT1 external interrupts. The temperature and pressure sensors' signals are connected to the inputs of the two ADCs, after passing through the signal conditional circuits, which are converted into digital values.



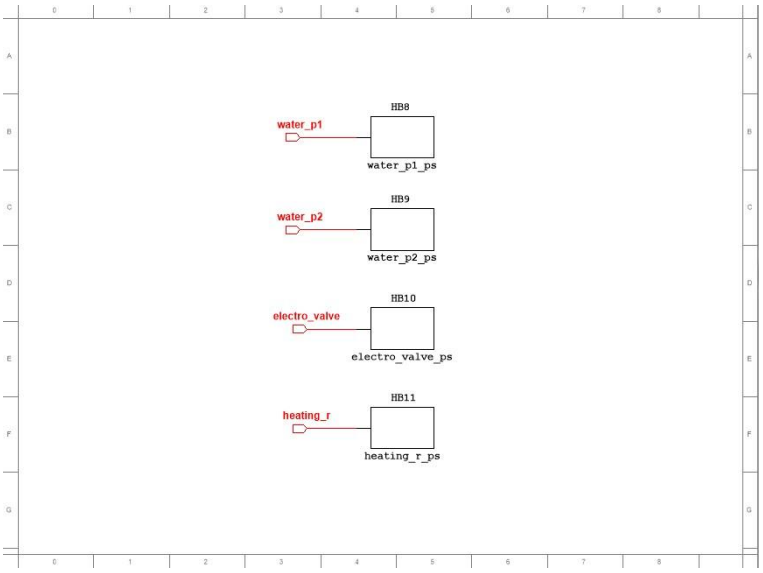
- **Serial/Logic Level Converter:** The logic converter is divided into two parts: the upper part connects to the Rx and Tx pins of the microcontroller and is also connected to is 5V VCC and ground, while the bottom part connects directly to the Rx and Tx pins of the Raspberry Pi, and to 3.3V and ground. This configuration ensures that the signals are properly converted between the two different voltage levels (5V and 3.3V) facilitating the exchange of data between the AVR microcontroller and the Raspberry Pi.



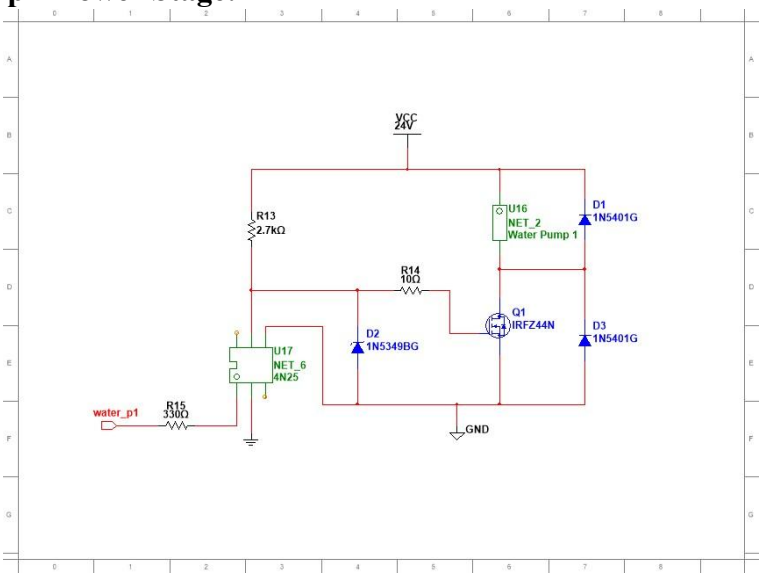
- **Capacitive Sensors**



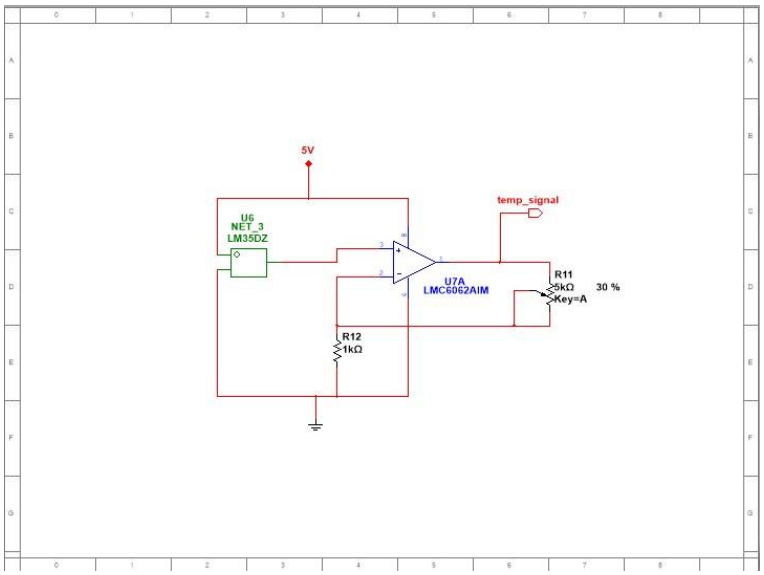
- **Power Stage:** The power stage circuit is designed as a chopper circuit. The PWM signal from the microcontroller is connected to the internal LED of the optocoupler. The ground of this LED is the same as the control section of the system. The transistor inside the optocoupler is connected such that the emitter is connected to the ground of the power section and the collector is connected to a resistor that goes to VCC (24V). A 1N5349 Zener diode connected between these two pins to act as a 12V voltage regulator to protect the MOSFET. The gate of the MOSFET is connected to the collector of the internal transistor of the optocoupler. Two 1n5401 diodes are connected to the MOSFET, one in parallel with the MOSFET having the cathode of the diode to the drain of the MOSFET and the anode to the source, and the other going from the drain to VCC in parallel with the load. These diodes protect the MOSFET and load from damage by preventing current from flowing back to them.



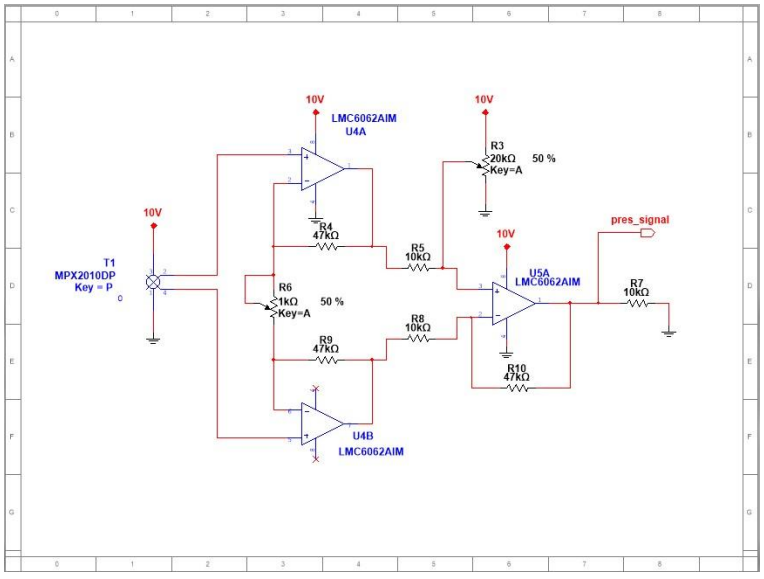
- **Water Pump 1 Power Stage:**



- **Temperature Sensor:** The LM35DZ signal conditioning circuit conditions the output of the LM35DZ sensor using an operational amplifier voltage buffer configuration. The noninverting input of the LMC6062 op-amp is connected to the output of the LM35DZ sensor, while the inverting input is connected to a 1k ohms resistor that is grounded. The loopback resistors of the circuit is a 5k ohm potentiometer that allows the gain of the circuit to be adjusted. This circuit has a gain of 5, meaning that if the output of the temperature sensor if 0.25V, the output of the circuit will be 1.25V.

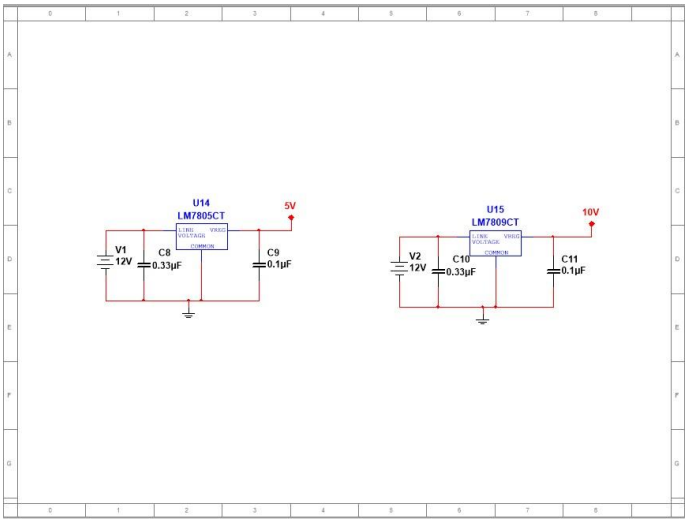


- **Pressure Sensor:** Three OP-AMPs are use to amplify the signal of the MPX2010 pressure sensor. The sensor's differential outputs are connected to the input of two different OPAMPS and their outputs are connected to the inputs of the other OP-AMP. The outputs of the sensors are connected to pin 2 and 5 of the instrumentation amplifier, with feedback resistors set at 47k ohms, and a gain resistor of 1k ohms. The offset resistor is 20k ohms, and the output of the instrumentation amplifier (pin 1 of the third op-amp) is connected to the voltage input of the ADC.

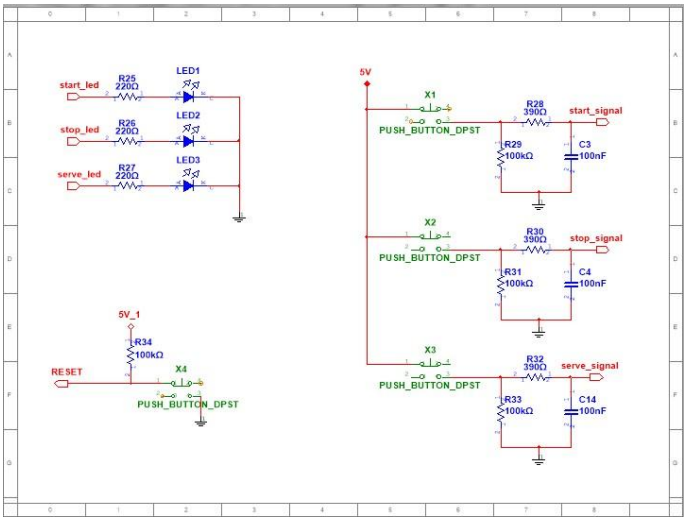


- **Power Supply:** The power supply circuit consists of two voltage regulators: LM7805 and LM7810, both of which are connected to the 12V output of the DC plug power supply adapter. A 0.33uF capacitor is connected between ground and input voltage, and a 0.1uF capacitor is connected between ground and output voltage for the 5V regulator, which

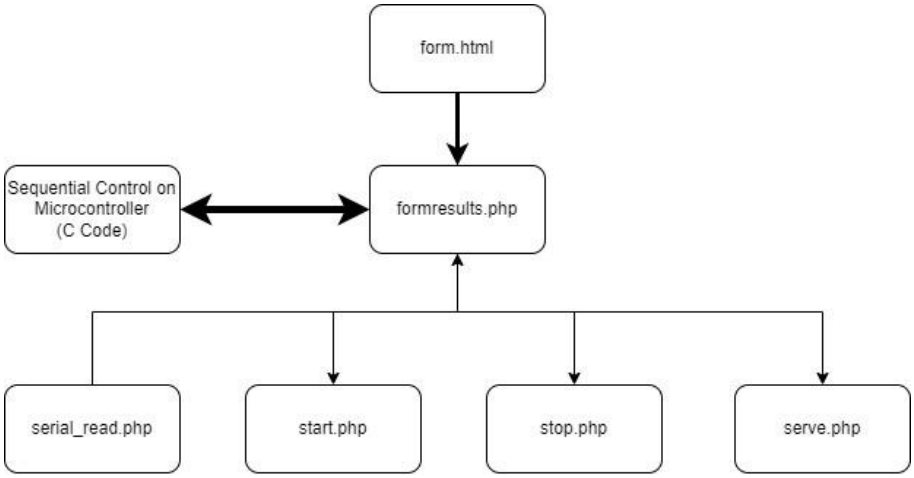
supplies power to the microcontroller, ADC, LM35 temperature sensor, and op-amp voltage buffer. These capacitors help to filter out noise and ensure stable voltage output. The same configuration is used for the 10V regulator which is supplies power to the pressure sensor and the instrumentation amplifier. This circuit is designed to provide stable and regulated power to the various components of the system avoiding damage from voltage fluctuations.



- **Control Panel:** The control panel consists of three push-buttons and three LEDs. The LEDs are configured with a simple circuit consisting of an LED and a 220 ohms resistor, which limits the current. The signals that turn the LEDS ON and OFF come directly from pins PB2, PB4, and PB4, for the start, stop and server signals respectively which are all configured as output ports. The circuit configuration for the push buttons includes a pushbutton, a 100k resistor to limit the current and protect against overvoltage, and a low-pass filter consisting of a 390 ohm resistor and a 100nF capacitor. This low-pass filter prevents switch bouncing by avoiding multiple transitions from happening during switch closure, which in return provides a cleaner signal.



7.0 Software Description



- **Sequential Control:** This code reads data from the two sensors via the ADCs. This code initializes the USART communication and sets up the control pins for the two ADCs. The ADCs are controlled by four control pins, and two external interrupts that are used to trigger when the data from the ADCs is available. When the code selects the appropriate /CS of the ADC, the microcontroller waits for the interrupt signal, reads the data, and deselects the sensor. After the data is collected, the temperature value is calculated, converted to Celsius**.** The pressure data is also calculated to obtain the level of the processing tank in millimeters which will be eventually converted to volume on the Raspberry Pi. The code also controls the water pumps, the heating resistor through pulse width modulation (PWM) signals based on the percentage input received via USART communication. The code receives input from USAR converts the numeric value to percentage for the PWMs duty cycle, and sets the respective PWM signal. Furthermore, to combine the sensors and actuators to work together, a sequential control has been implemented on this code based on the stop and start signals, and the state of the sensors and actuators. Each step represents a particular task in the operation of the system that involves both the sensors and actuators. The system waits for the user input or sensor values to advance to the next step. This system uses the Rx interrupt function to read incoming bytes whenever new data is received on the USART interface. The commands that used in the first byte are hexadecimal codes that represent the different types of data, the second byte represents the data being received. These commands are predefined so that the microcontroller and the Raspberry Pi can interpret the data correctly. These are the commands that are being used to read serial data on the Microcontroller:

    - **0x5B:** Water pump 1 received speed data
    - **0x4A:** Tank 1 volume data
    - **0x3D:** Water pump 2 received speed data
    - **0x2C:** Tank 2 volume data

- **0x1F:** Liquid's temperature data
- **0x4E:** Heating resistance power data
- **0x0E:** Cooldown timer data
- **0x0A:** Start signal
- **0x5F:** Stop signal
- **0x1B:** Serve signal

- **form.html:** This HTML code defines the parameters for the system. It has multiple input fields that are supplied by the user, including one for liquid percentage from each storage tank, two for water pump speeds, one for the temperature, one for the heating resistor power, and one for cool-down time. This code also uses a JavaScript function that checks if the sum of the two inputs for the liquid percentage parameters is less than 90%, and alerts the user if the value is higher than this limit. All the inputs are required, and the ranges are checked using parameters. This form submits to the formresults.php PHP file.

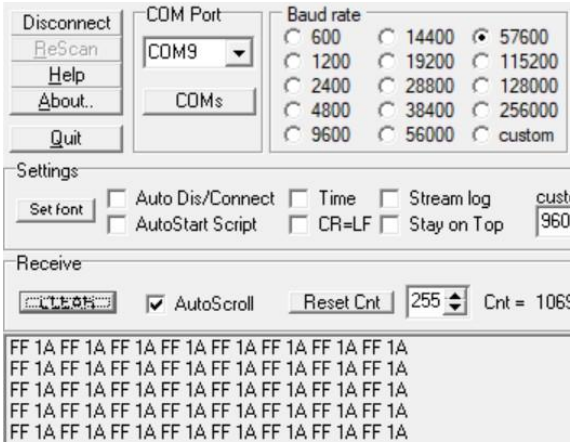
- **serial_read.php:** This PHP script reads data from the serial port sent by the microcontroller. It reads one byte at a time looking for a command byte, and uses a switch statement to process the byte according to its value. Depending on the value of the command, the script assign different values to the variables. The script stores the received data into an array and returns it the formresults.php file as JSON. These are the command values being used to read values from the microcontroller:

  - **0xA3:** Step ○ **0xB4:** Temperature
  - **0xC5:** Volume
  - **0xD1:** Start state
  - **0xE2:** Stop state
  - **0xF6:** Serve state
  - **0x9A:** Cooldown counter

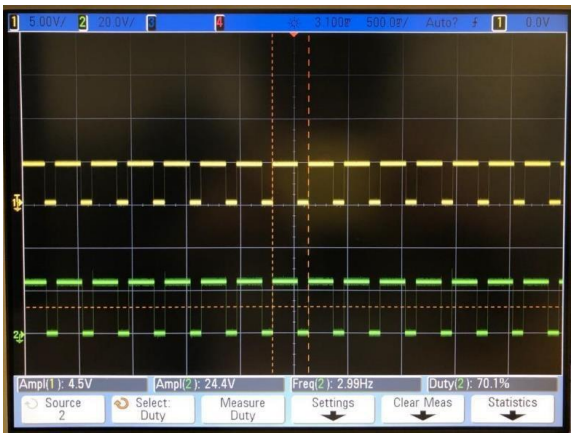
- **start.php, stop.php, serve.php:** These scripts transmit the start, stop, and serve signals to the microcontroller via serial communication using the 0x0A, 0x4A, 0x1B bytes respectively, followed by the data values.

8.0 Testing and Calibration

- **Testing:**
  - **Sensors Code**: These are the values obtained by the ADCs, the 0xFF value indicates that the processing tank has reached maximum capacity as determined by the pressure sensor. Additionally, the temperature sensor's converted value is represented by in Celsius, which is equivalent to 26 in decimal, and since this test was done at room temperature, this outcome reflects the effectiveness of the conversion process and proves that the temperature sensor and its corresponding code are operating as expected.



  - **Actuators:** Due to the chopper circuit configuration, the PWM signal has to be inverted to function properly, meaning that the duty cycle seen by the load is 30%. The signal of the first probe is the input of the optocoupler, and since the circuit was being tested, a signal generator was used to simulate the PWM. The signal on the second probe is the output of the chopper circuit, and it can be observed that the MOSFET is working as expected having an output of 24V, and modulating the same square wave as the optocoupler's input signal.



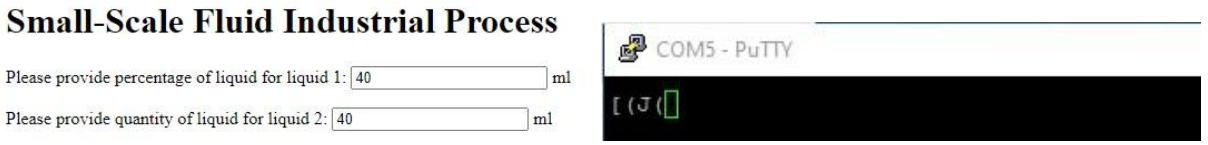
- **Water pumps:** The following table shows the performance of the water pumps under different duty cycle percentages:

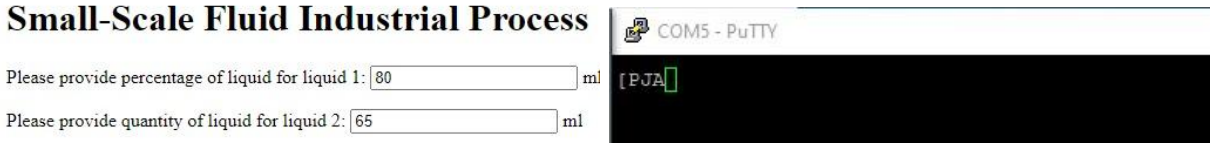
Duty Cycle %	Current (A)	Voltage (V)	Power (W)
10	0.07	1.1	0.077
20	0.25	3.7	0.93
30	0.27	6.3	1.7
40	0.32	8.6	2.75
50	0.47	11	5.17
60	0.62	13.3	8.25
70	0.76	15.6	11.86
80	0.9	18	16.2
90	1.08	21	22.69
100	1.17	23.6	27.6

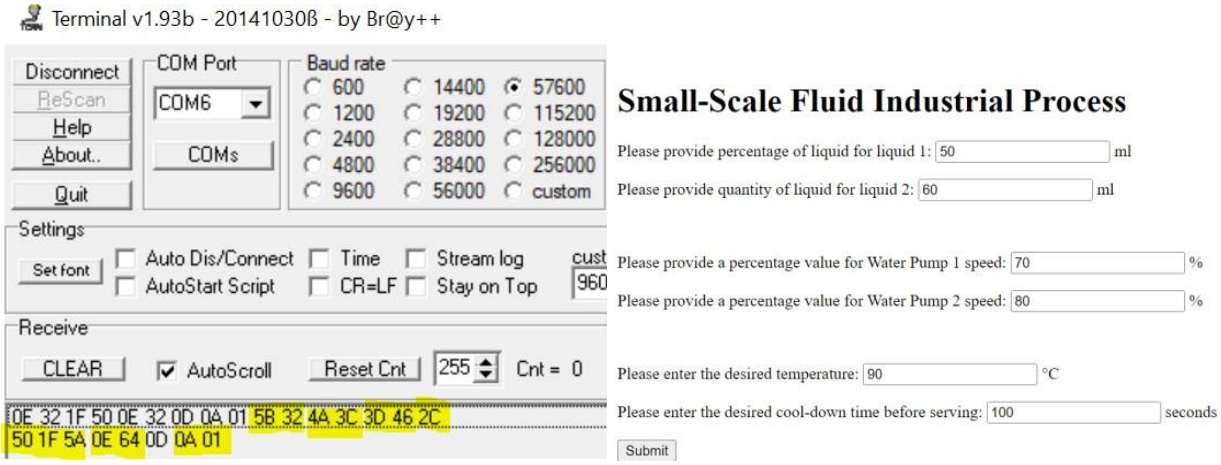
- **Heating Resistance**: The following table shows the performance of the heating resistance under different duty cycle percentages:

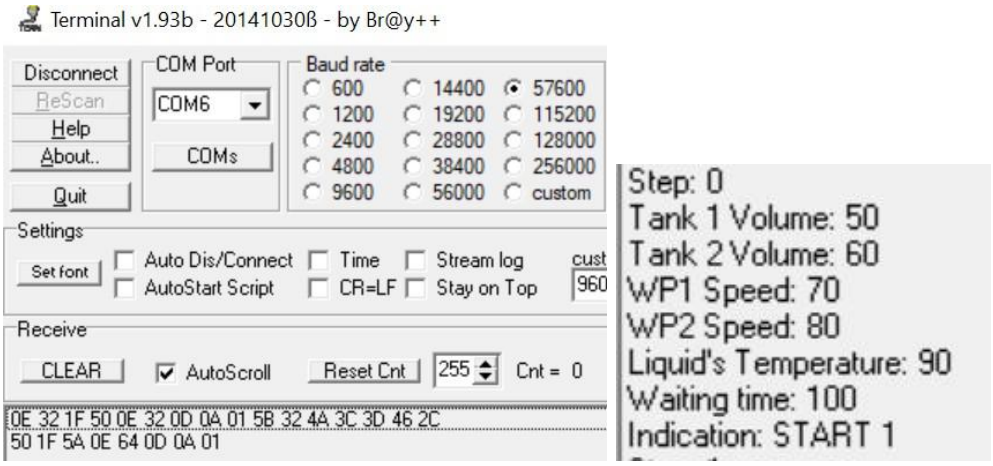
Duty Cycle %	Current (A)	Voltage (V)	Power (W)
10	0.07	1.94	0.14
20	0.18	4.65	0.84
30	0.26	6.94	1.80
40	0.34	9.20	3.13
50	0.42	11.46	4.81
60	0.50	13.74	6.87
70	0.58	16.03	9.30
80	0.66	18.33	12.10
90	0.76	21.30	16.19
100	0.82	22.9	18.78

- **2-byte configuration being sent by the web server:** These are test examples of how the microcontroller receives the data values. I had my PC connected as a "spy" to be able to see the values being sent by the microcontroller. As we can see in the second example, the Water pump 1speed percentage is 80% (I had made a mistake when I took this screenshot and put percentage of liquid in the html file instead of water pump 1 speed percentage), and the command corresponding to water pump 1 speed percentage is 0x5B, and 80 in hexadecimal is 0x50, which corresponds to "[P" in ascii

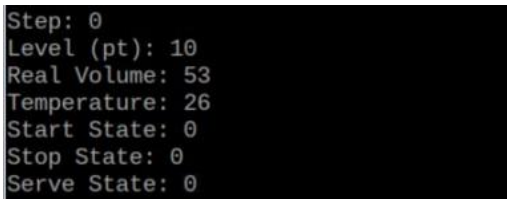




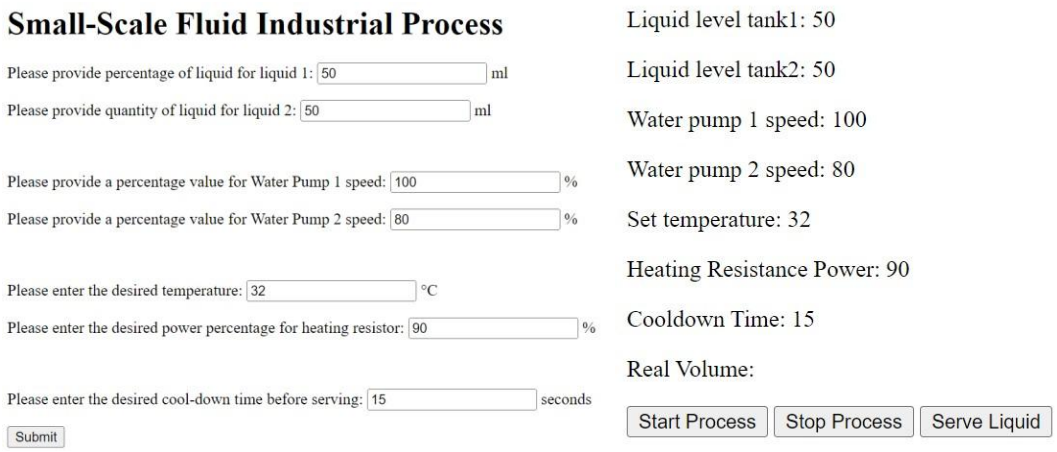




- **Sequential Control:**
  - Step 0: The microcontroller is waiting for the values from the web server. We can see that the volume of the tank is less than 100ml. and the status of the start button is 0.



  - **Step 1 and 2 :** The web server sends all the parameters values and the start signal, and since the processing tank is has less than 100ml, it goes directly to step 2. In step 2, water pump 1 starts. The maximum volume of the processing tank is 1300ml, and since the parameter value for the percentage liquid from storage tank 1 is 50%, water pump 1 will be ON until the pressure sensor detects that the volume in the processing tank is around 750ml.



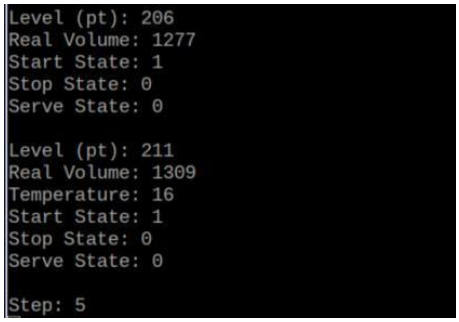
```
Step: 0
Step: 1
Step: 2
Temperature: 26
Start State: 1
Stop State: 0
Serve State: 0
```

- Step 3: When the desired liquid volume from storage tank 1 is transferred to the processing tank, water pump 1 stops.

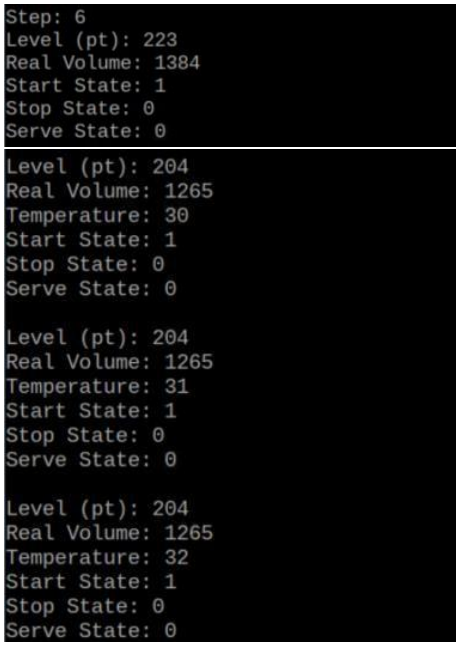
```
Level (pt): 104
Real Volume: 640
Temperature: 29
Start State: 1
Stop State: 0
Serve State: 0

Level (pt): 107
Real Volume: 659
Temperature: 29
Start State: 1
Stop State: 0
Serve State: 0

Step: 3
Level (pt): 107
Real Volume: 659
Temperature: 29
Start State: 1
Stop State: 0
Serve State: 0
```

- Step 4: After water pump 1 stops, the sequential control will go to step 4 as long as all the conditions are met, and water pump 2 starts and will continue to stay ON until the water volume in the processing tank is around 1300ml because the user defined the volume percentage from storage tank 2 as 50% as well.

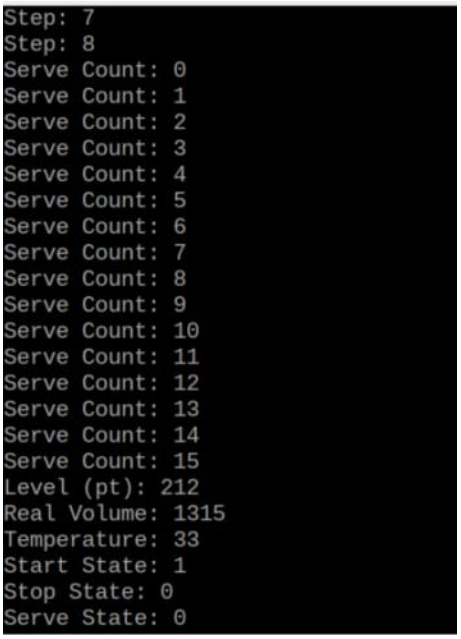
```
Step: 4
Temperature: 30
Start State: 1
Stop State: 0
Serve State: 0
```



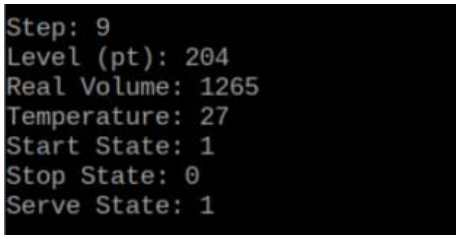
- Step 5 and 6: Once the volume is reached, water pump 2 is stopped in step 5, and in step 6 the heating resistance is started with the percentage power provided by the user. This resistance will stay on until the desired liquid temperature is reached, in this case, 32°C



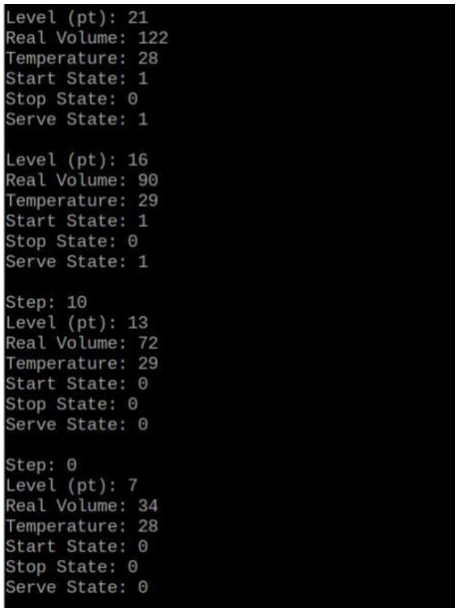
- Step 7 and 8: In step 7 the heating resistance is turned off, and in step 8, the cooldown counter starts, it will count until 15 in this case. Then, the code will only go to step 8 if the user sends the serve signal.



- Step 9: Here, the electro-valve is opened, and it will reman open until the liquid level is below 100ml.



- Step 10: In step 10, the electro-valve is closed, and the sequential control goes back to step 0



- **Calibration:**
    - To calibrate the pressure sensor, the offset voltage must be set to 1V when the processing tank is empty, and to 4V when the processing tank is full (1300ml). ○ To calibrate the temperature sensor, the gain must be 5. In other words, for example, if the output of the temperature sensor is 0.23V, the gain potentiometer has to be increased until the output of the OP-AMP is 1.15V.

8.0 Troubleshooting

Multiple test points were placed on the PCB to troubleshoot the most important signals:

- As mentioned previously, if temperature readings are erroneous, check the output of the sensor, if it is not around room temperature in millivolts, replace sensor. If the output of the sensor is as expected, check the output of the OP-AMP, if the voltage does not have a gain of 5, calibrate using potentiometer, if you are not getting any gain, replace OP-AMP.

- For the pressure sensor, check the output of the third OP-AMP. When the processing tank is empty, the output of the OP-AMP should be around 1V, if not calibrate using offset potentiometer. Next, fill the tank with 1300ml of water and check the output of the OPAMP, if the output is not around 4V, calibrate using the gain potentiometer. Empty tank and check voltage, it should be around 1V, if not calibrate using offset potentiometer, repeat this process until the output of the OP-AMP is 1V when the tank is empty, and 4V when it is full. If the pressure sensor is not working at all, check the output of the 7810 regulator, if the output is not close to 10V, replace regulator.

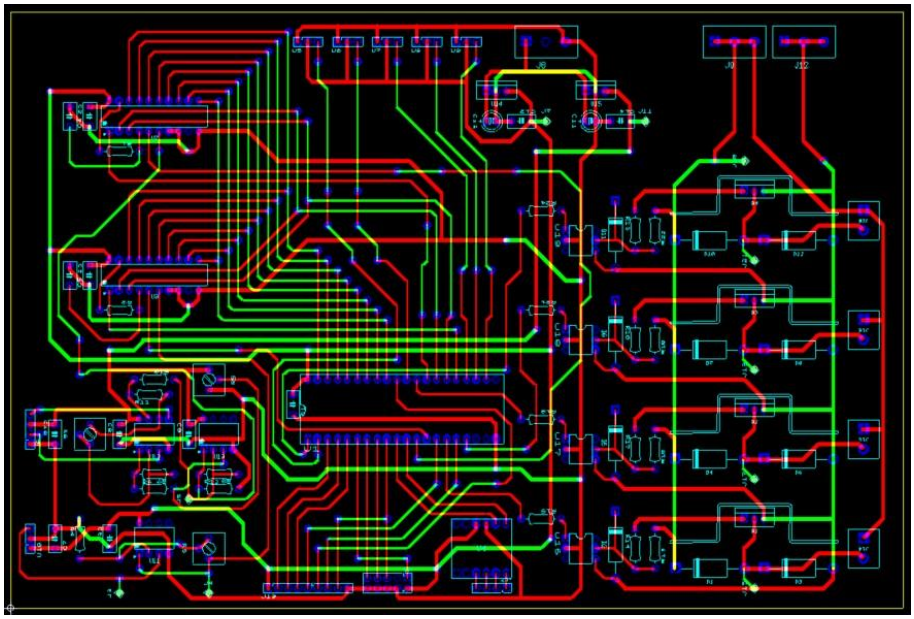
- If actuator is not working properly, check the optocoupler's input voltage, since the signal is inversed, when the actuator is ON, the voltage should be less than 5V, depending on the PWM. If the input of the optocoupler is not close to 0V when the actuator is working close

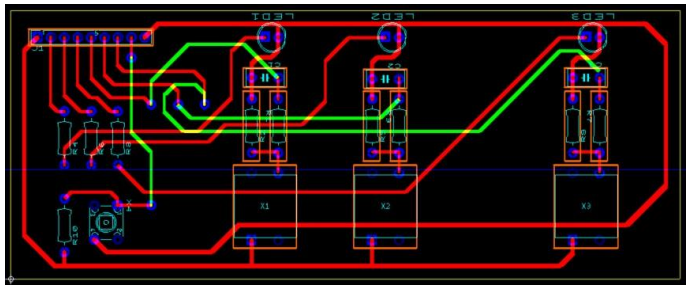
to its maximum power, replace microcontroller because PWM is not working properly. If the input of the optocoupler is the voltage expected, check the drain of MOSFET, it should also be close to 0V, if not, replace MOSFET. Another solution is to check the output of the 24V power supply, if the voltage is not close to 24V, replace power supply.

Appendix

A. Illustrations

- **PCB: Layout:**





B. Bill of Materials

Quantity	Description
4	2.7kΩ RESISTOR
4	IRFZ44N
4	10Ω RESISTOR
4	1N5349BG
8	1N5401G
4	4N25
4	330Ω RESISTOR
2	Water Pumps
1	Heating Resistance
1	Electro-valve
2	HDR1X9
1	LM7805
13	0.1µF CAPACITOR
1	LM7810
3	HDR1X3
2	0.33µF CAPACITOR
3	47kΩ RESISTOR
5	10kΩ RESISTOR
1	1kΩ POTENTIOMETER
1	20kΩ POTENTIOMETER
3	LMC6062
6	Capacitive Sensors
1	5kΩ POTENTIOMETER
1	1kΩ RESISTOR
1	Logic Level Converter
1	HDR1X4
2	ADC0804
2	150pF CAPACITOR,
1	HDR2X5
1	Microcontroller, ATmega8515
3	220Ω RESISTOR
3	LED
4	PUSH BUTTON
3	390Ω RESISTOR
4	100k Ω RESISTOR

C. Cost Analysis

Component	Quantity	Cost Per Unit	Total
MPX2010	1	$39.35	$39.35
SEN0508	5	$15.70	$78.50
LM35DZ Module	1	$9.95	$9.95
AN7805	1	$0.65	$0.65
AN7810	1	$1.68	$1.68
ADC0804	2	$8.28	$16.56
ATmega8515	1	$6.69	$6.69
Water Pump	2	$41.55	$83.10
Electro Valve	1	$13.79	$13.79
Heating Resistance	1	$2.91	$2.91
MAX232	1	$3.84	$3.84
½ Vinyl Tube	1	$14.95	$14.95
¼ Vinyl Tube	1	$17.25	$17.25
LMC6062	4	$4.54	$18.16
4N25	5	$0.92	$4.60
		Total: $311.30	