Continuous Chemical Reagent Mixing Automation

Amanda Gilliam, Anish Umashankar, Ernel Reina, and Jason Scislaw

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — In Chemistry, continuous mixing can react reagents more efficiently than batch mixing. In partnership with Helicon Chemical Company, we developed a system to convert their operations from batch mixing to continuous. Helicon built the system, and we developed an automation upgrade to their system. The upgrade interprets user inputs to operate the mixing pumps, route the chemicals via closed loop servomotor control, and log process statistics.**

***Index Terms* — Manufacturing automation, Control engineering, Centralized control, Closed loop systems, servosystems, microcontroller**

I. Introduction

We approached Helicon with the idea to automate their system for our project because it was challenging, it was closely connected to the industry projects we would see in our careers, and it was a system that could be adopted in a company's critical operations. Helicon was motivated to work with us because we offered a cheap solution to address short term needs, and they were under no obligation to implement our system if it didn't meet their specifications.

The automated solution consists of a centralized controller that receives input from the user on several control parameters. The controller uses these parameters to operate two peristaltic pumps by varying the pump speed and runtime. Before and after the mixing process, the controller operates servomotors mounted to mechanical ball valves to change the system routing between 1) the two reagent lines, and 2) inert gas and vacuum lines. During the process, the system continuously monitors for errors via built in sensors and enacts an emergency shutdown procedure if conditions are suboptimal.

The primary objective for this project was to design and build a functioning system that automates Helicon's process, with Helicon's *adoption* of this system a secondary goal. In order to still provide value to Helicon in case this secondary goal wasn't met, the system was designed with modularity in mind so that Helicon could easily tweak the system to meet their needs. Other objectives were to maintain secrecy of Helicon's proprietary information and maintain compliance with industry and government regulations that Helicon are obligated to.

II. System Components

In order to meet the requirements as well as overcome design hurdles we first had to choose proper components for each. These components were chosen with three primary concepts in mind: functionality, cost, and ease of use.

A. Pumps

The pumps we had to interface with were already chosen by helicon prior to the project. So the design hurdles behind the pumps was simply interfacing with them and getting them to work with our microcontroller and having the pumps sync with the rest of our system through it.

First we need to understand how the pumps work mechanically. Two peristaltic pumps are used to control the flow of the reagents. A peristaltic pump has a rotor with rollers attached to the ends, shown in Figure 1, and tubing is threaded through a channel along most of the rotor's circumference. As the rotor rotates, the rollers pinch the tubing, driving material through in the direction of rotation. The material's flow rate is directly proportional to the angular velocity of the rotor. From the perspective of process control theory, the pumps in Helicon's system are the control elements in a control loop driving the flow rate variable of the pump system.

Now that we understand how it works we know what our microcontroller should try to accomplish. Next is how to interface with the Pump. The peristaltic pumps

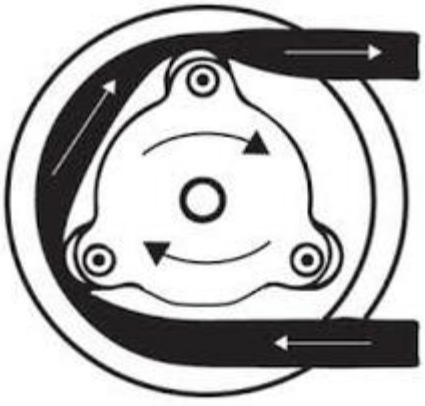


Figure 1- Peristaltic pump: basic schematic

currently implemented in Helicon's system have their own independent controllers. These controllers rely on human input however and must be modified or bypassed in order to control the pumps as part of our project. The pumps come equipped with DB25 female connectors with pins dedicated to third party control signals. These pins read an analog voltage from 0-20mV, or with an adjustment, 0-10V, and internal hardware translates that voltage to a flow rate. Our controller will be working with digital signals, so our system will need to implement digital to analog conversion. These D-A converters use pulse width modulation (PWM).

So using 8 pins for each pump with PWM capabilities from our microcontroller we were able to get a clean reading of the output and control the pump's speed.

B. Valves and Motors

The valves for this project were also chosen prior to the project and are instead an object to design around. Testing showed that the valves would need about 0.435 ft-lb of torque to move. Due to their structure and the requirement placed by helicon for ease of removal, our method to move these valves had to be one that used pressure or latched on to them rather than replacing the valves or welding on instruments. For this reason we chose to use couplers with pressure screws.

In addition to this the valves had to be able to operate in three different positions. To solve this we came up with tow solutions both of them being motors of different types. Stepper motors and Servo motors. A stepper motor works much like a servo motor in that it has the ability to move discrete steps and accurately position itself. These steps are outlined on the product description in the form of degrees per step. However A stepper motor has a much larger amount of poles as a servo (50-100 versus 4-12). Each pole is a north or south magnetically generated pole and is what allows motors to spin. Stepper motors don't need encoders to determine their current position thanks to the large amount of poles they have. Stepper motors instead move 1 step at a time whenever they get pulse. A stepper motor on the other hand only requires 1 input/output pin to operate and gives up precision for power. Since we really only needed the valves to be in one of 3 positions precision was not a large concern. The steppers we found and even tested were not strong enough to move the valves and would have required extra gears and tinkering to get working. So instead we decided on the servo motors. They met the torque requirement and we chose servo motors with 270 degree of motion which was enough to meet the 3 position requirement.

C. Controller

The ARM AMD series of processors that are designed for low-power high efficiency projects such as ours. Some of the advantages they provide is they are very cheap compared to other processors. This paired with the fact that, compared to TI MSP430x series microcontrollers, they have less I/O pins allows us the flexibility of purchasing two ARM AMD series processors for around the price of one MSP430x series microcontrollers. The inclusion of two ARM AMD series processors allows us to create a master-slave relationship between the two processors, which provides ease of programming to one microcontroller while controlling two. This effectively negates the downside of not having enough pins as we can take the I/O pin counts of both microcontrollers and add them together.

The two ARM AMD series processors that we have taken into consideration are the ATSAMD21G18A-MU and the ATSAMD11D14. Our choice for these two processors stems from the fact that these two processors provide all the needed specifications such as adequate memory, I/O ports, frequency, power, communication type, and cost. In addition, these two processors are the same two processors that are used in the Arduino MKR family boards, ZERO and motor carrier. In terms of function, these two boards provide exactly what we are looking to do in Helicon Chemical Company's existing pump system, which is to automate motors, log data, and control sensors. These two boards are made to work together and for this reason they will serve as a good starting point in terms of design and prototype testing. Also they are programmable using Arduino's IDE and exist within the Arduino development community which is filled with information, libraries, tutorials, and open-source code all found online for us to use or reference if/when we need it.

The amount of I/O pins actually was not enough with these two alone so we decided to include a port expander, the MCP23017ML, which connected to our microcontroller via I2C and was enough to meet all our I/O needs and have a few left over for expandability purposes.

D. Enclosure

The enclosure that will house all our delicate equipment has to be able to possibly withstand dust, lightly splashes of liquids, and possibly corrosive material. To this end we had the options of getting metallic or nonmetallic structures. These can be stainless steel, Polycarbonate, fiberglass and a few others. Of these, fiberglass enclosures and polycarbonate enclosures

comply with NEMA standards for being resistant to corrosion or non-corrosive respectively. The cheaper option is the polycarbonate one and seemed like the best choice for our project. It can even have a see through cover so technicians can better assess if something is going wrong internally.

With this in mind we used a 10x12x6 fiberglass enclosure box standardized with Nema 4x so it can resist splashes of water or other liquids and is also non corrosive in case of an accidental spill of hazardous material. This enclosure will have to be adjusted to allow us to mount all the different components placed upon it, such as the push buttons, dials, and LCD.

Internally it will have to house DIN rails to mount our circuit breaker, power supply and Terminal blocks. Also inside our PCB board and buzzer for the alarm system can be found.

III. Implementation Strategy

In order to maintain our non-invasive goal for helicon we want a system that works with all the components already at hand. Below, Figure 2 shows our project design overlaid with the current Helicon system. As noted in the legend, Helicon's system is in blue, our system is in black. The main components/parts of our project are represented in 'block' form to show the flow of hardware and software feedback loops that our project will contain The blue section, showing Helicon's system, shows the some major components already talked about with the other two being the reagent tanks which store the chemicals for the reaction and the argon tank/vacuum pump. Helicon's reagents are air sensitive. As air will be introduced after each reaction during cleaning, the system will need to be purged with inert gas to remove reactive elements. The Argon tank and vacuum pump will perform this function. Our system as it's currently proposed will not interact directly with these components.

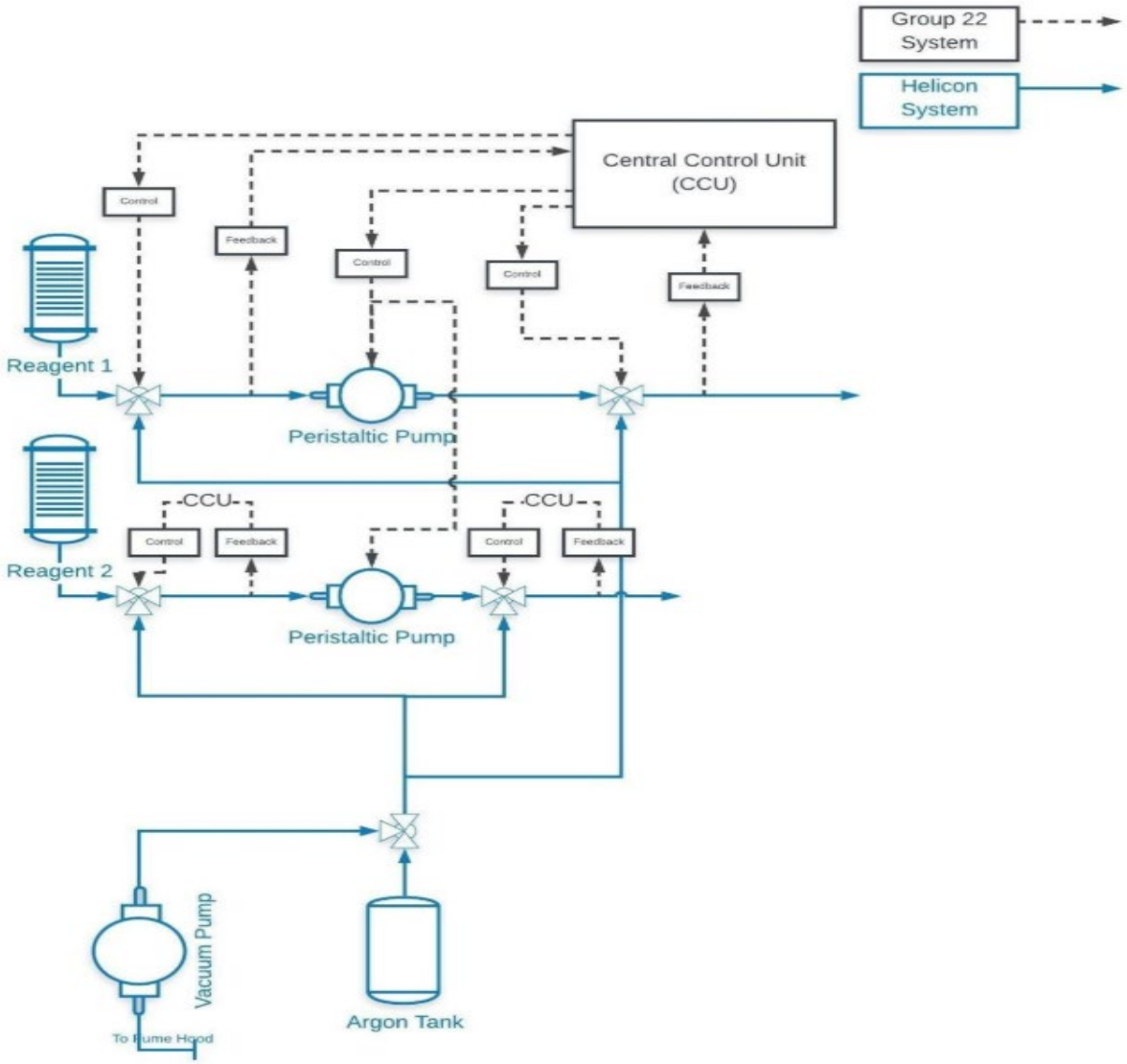


Figure 2 - Diagram of Helicon's Pump system (Blue), and our Automation System (Black)

A. Power

The table below lists our projected maximum power consumption.

Model	description	qty	Volatge (VDC)	Amp (A)	Power (W)
DS3225MG	Servo Motor 25kg-cm 270 degree	4	5	2.5	50
TC1602A-01T	LCD screen for arduino	1	5	0.15	0.75
ABX00012	MKZero Microcontroller	1	5	0.02	0.1
ASX00003	Motor Carrier board	1	5	0.05	0.25
	LED	5	5	0.02	0.5
	Buzzer	1	5	0.5	2.5
Total			5	10.82	54.1

We decided to try and operate everything under the same voltage so we could use just 1 power source and not have to complicate our design trying to get different voltages out of said power source. Now the power Helicon would have available is typical house-hold power of 120VAC and would have to be converted. To this end we implemented a typical power chord to enter our enclosure, from here it enters some terminal blocks before being passed through a circuit breaker and finally reaching our power supply. This power supply transforms our AC power to a 5VDC, 10Amp source. The above mentioned 54 Watts is with the idea that all four motors are operating simultaneously and at full power which is not realistic so this source was enough to power all our devices.

IV.RELATED STANDARDS

The following are standards related to our project which had an impact on how we carried our project along including testing and implementation.

First is the IEEE 829 - Software Test Documentation. As the name of this IEEE standard implies, this standard focuses on the meticulous documentation of any and all experimental activities regarding software testing. This standard specifies a set of documents (divided into 8-10 defined steps/stages) that each have their own set of documentation. In order for this standard to be used for any type of software or system testing environment, it specifies each document does not have to be produced and that there is no grading rubric regarding the information inside of the documents; the judgment of these falls upon the person or group following this standard. The following are the steps that we as a group followed/documented and our justification for its importance:

Master Test Plan (MTP): This stage is necessary as it provided a generalized testing plan for our entire project and will have basic level documentation on all stages of testing, we pursue.

Level Test Plan and Design (LTPD): This is a combination of two levels documentation listed in IEEE 829, but we believe combining these levels will streamline our testing and overall documentation process. This level documents everything that is happening in our testing process. This includes, but is not limited to overall approach, all resources being used, components to be tested, and a general schedule of all the aforementioned tasks and when they are being done. In addition, this documentation will include a list of all the specific test cases along with their respective criterion, procedure, and ONLY our expected results (observed results will be in the Level Test Log documentation).

Level Test Log (LTL): This set of documentation is very straightforward, and will include most, if not all, of the data we gathered during testing. This documentation will also have a section specifying what problems we have in our data.

Master Test Report (MTP): This documentation serves as the conclusion of our set of documents and being so, summarizes all of the previous test documents and highlights all of the key information listed. The MTP should also serve as a 'resume' of sorts that we could show as a final document of our entire testing process.

The impact of IEEE 829 towards our project design is minimal. The IEEE 829 standard only specifies having thorough testing documentation and because of this, it has very minimal impact towards our actual design. The only possible impact that this could have had to our design is in the hypothetical situation in which we have a working design in place and due to our thorough testing and documentation listed by this standard, we discover a fault in our design and from all the testing documentation we can go back into our design phase and correct the error(s). This standard serves as only a set of documentation guidelines towards our software, it does not present guidelines to actually create a working design.

Next is the Nema standards for our enclosure. The National Electrical Manufacturere's Association standards for electronic enclosures are meant to clarify concepts like waterproof, sealed, and dust free. These standards should be consulted for our enclosure. NEMA also rates motors these are for rating its frame size, efficiency, testing, and operations. This is relevant for stepper motors which were a strong candidate in our design.

The main impact of these standards is for us to decide whether we need our enclosure that houses delicate electronics to be resistant to things like hose directed water, or corrosion. The enclosure is physically close to chemical processes which could result in a fire taken out by a sprinkler system or in a spill of a corrosive chemical onto the enclosure itself. Since the likelihood of these situations is somewhat possible we seriously considered using an enclosure that was capable of withstanding such accidents so that helicon would not lose the control system as well. For our enclosure the most prevalent standard/enclosure choice was one rated for Nema 4x which provides protection against windblown dust and particles, splashing water, and corrosion, all of which prove useful for our design.

Lastly, Helicon Chemical Company resides within a UCF affiliated business incubation space, therefore our design must also take into consideration UCF Health and Safety Standards. There are many standards within UCF's code of standards such as radiation, laser, workplace safety. Fortunately for us, the type of work that Helicon Chemical Company is involved in does not involve radiation and lasers, therefore these health and safety concerns can be ignored. In addition, due to the fact that we will not be operating the automated pump device that we have been tasked to create and the fact that we are not employed by Helicon Chemical Company, there is only so much we can do on our end to ensure that workplace safety is met. This involves specific development of our PCB enclosure as it houses all of the potentially dangerous electronics.

Unlike UCF workplace safety, UCF fire safety concerns must be acknowledged and taken into consideration when designing our final product. Despite the fact that the environment that our pump automation cart will reside in is already properly situated for any fire or workplace issue, we must also attempt to mitigate all possibility of fire hazards within our product. The specifics of UCF's fire safety policy involves many facets, most of which do not apply to Helicon Chemical Company or our device. These include fire safety regarding forest fire and other outdoor fire safety precautions. In our case, the only two precautions to note are emergency protocol regarding fire safety (this is already in practice at Helicon Chemical Company in their emergency fire safety protocol) and NFPA accordance. The NFPA is the National Fire Protection Association and so long as our electronics are thoroughly tested and operate within their given operating conditions (found in electronic specific datasheets) then we will have been in accordance. To avoid any complications and potential hazards in design, we will be prototyping our design from the Arduino MKR motor carrier and MKR zero boards which have been thoroughly tested and vetted by Arduino, as they are consumer available and ready products. Just in case, we tested the products even more as a precaution.

The main impacts of these standards are enclosure and testing related. In terms of enclosure, to avoid any electronic hazards from directly interfering with the chemicals that are involved in our system and the operator of the system, we must design an enclosure that houses all our open electronics. In addition, this design must be electrically and thermally stable so that in the event that our electronics were to overheat, spark, or break the enclosure will prevent any further damage to our system by blocking off direct contact to the chemicals and the user.

V.SOFTWARE DESIGN

The controller is the brains in the control loop. The controller needs to calculate the instantaneous error, the accumulated error, and the rate of change of error at any given moment. With this information, the controller decides a corrective action to take and generate a signal to communicate that action to the control element. For the control loop subsystem, the controller module will overlap with the CCU subsystem and it's software and hardware will be part of the processor module.

One of the benefits to our choice of microcontroller is that we can utilize the Arduino IDE. This makes it easy to program the chip via USB since the IDE has a built in upload button. Of course, the IDE also provides a compiler for the code.

The Arduino IDE uses its own Arduino language which is based on C/C++ with more object oriented class structures. This made it easy for our team to use since we are all familiar with C. The Arduino language is composed of three main components, which are functions, variables, and structure. The structure component is also both familiar and not familiar to our team, as it contains the familiar logic statements like 'if' but also two required functions to make a 'sketch' work. These two functions are loop(), which takes the place of a main, and setup(). Loop() is repeatedly run without any necessary code on the programmers part. This is where the bulk of our code went. We put all of our startup code that will be run once on boot, such as pin initialization, within the other required function setup().

Another added benefit of using this language is that there are already many libraries that come included with the IDE as well as community contributed libraries available in public repositories like github. This made it

very easy to interact with the peripheral systems we are using, such as the SD card and LCD screen, which saved time during development.

A. Pump Control

The pump for starters will need the following Inputs: Remote Start/Stop (Digital), 0-10V Speed Control Input (Analog) and wont use the following inputs: Remote CW/CCW, Remote Prime, Aux in. The peristaltic pump inputs work with current sinking outputs, through NPN transistors with open collectors, or with contract closures to earth ground. To start and run the pumps, a continuous low signal is sent to the Remote Start/Stop input. The pump flow rate is controlled via analog input signals. There are two options: 4-20 mA analog signal, with 4 mA being Stop and 20 mA Full Speed, or a 0-10 V analog signal, with 0 V being Stop and 10 V Full Speed. Both options offer 10 bit resolution. For this project we used the voltage inputs and outputs. Although our current design does not plan to use all inputs, it's important to understand them in case they are required in the future. For example, if Helicon determines their system requires the pumps to run in both directions, this can be done with the Remote CW/CCW (Clockwise/Counter Clockwise) input. This input can be pulled to active low to run the pumps counterclockwise. The pump will slow to a controlled stop before changing direction. Secondly, Helicon has talked about needing to prime their system by running each pump individually until the two reagents are right on the edge of the mixer. This can be done with the Remote Prime input by sending a continuous active low signal.

Outputs used: 0-10V Speed Feedback Output (Analog), General Alarm Output (Digital)

Outputs unused: COM (Motor Running), Tach Output, Local Remote Indicator.

The peristaltic pump uses built in feedback mechanisms to measure the flow rate and converts this to an analog output signal. Similar to the input signal, the pump outputs both a current and a voltage in the same ranges with 10 bit resolution. For the purposes of this project this satisfies the feedback section of the flow rate control loop. Along with the speed feedback, we will use the general alarm output signal, which will communicate when the internal pump circuit detects any errors. This way our system will be in tune with the pumps and our controller can respond to any alarms by notifying the Helicon team via our alert system, or by initiation emergency shutdown.

It's important to note that the Alarm Output uses an open NPN collector. This output gives a "low impedance" state at earth ground and is essentially floating when in "high impedance" state. What this means is that extra care must be taken in wiring this output to avoid damage to external equipment. It is recommended that we use a current limiting resistor to avoid current surges at the low impedance state.

As with the inputs, our current design does not plan to use all the pump outputs. Still, it's important to understand them in case they are required in the future. The COM output shows whether or not the motor is running, which can add redundancy to our feedback system. The Tachometer output shows the RPM of the pumps, which Helicon could find useful for data collection. The Local Remote Indicator output gives a visual indication that the pumps are being configured/operated remotely. This could potentially be useful in the future if Helicon implements a procedure where the pumps are controlled both manually and remotely.

With the inputs and outputs outlined, it's also important to note that the MKRZero uses 5V logic, which will be incompatible with the 0-10V Pump motor speed input/output signals without modifications. Since both sides use 10bit conversion, the resulting circuitry is simplified. When the speed control signal runs from MKRZero ------> Pump, the 0-5V signal was amplified by two to access the full range of speeds offered by the pump use.

B. Ball valve automated control.

For the four loops governing flow direction. The control elements are manually controlled ball

valves that switch flow direction between two inputs to a single output. The ball valve will be modified so that a servo can be attached.

The motor needed relatively high torque, and needed to work every time. We purchased an adapter to couple the servo and the rectangular knob of the ball valve. Also the servo was securely mounted over the valve to ensure efficient and consistent force transfer. This mounting solution is semi-permanent, and the servo is removable so as to follow our requirements.. If our system fails at any time, Helicon must be able to convert their process back to manual operation.

To this end we had selected a NEMA 17 stepper motor with a 27:1 planetary gearbox allowing for up to 300 Newton Meter torque. However we quickly found this to be insufficient to turn the valves. After further testing we would need at least 0.435 ft-lb in order to move the valves. And so we switch to a servo motor capable of 1.8ft-lb.

The motor shaft and valve was secured in place through pressure by tightening the prisoner screws on the coupling. There were two concerns with this design idea. The least troubling was whether or not we could apply enough pressure with these to make sure the two pieces it

adjoined did not slip. If luckily we did not have to look to soldering. The other concern was the opposite problem which was whether too much pressure was applied to the valve side and ruins the threading which helicon currently uses. This ended up not being an issue either at least this far. Instead the issues which actually came up was that Helicon had their valves hooked up with tight wraps and our system needed secure installation onto the cart to make sure the motor was turning the valve and not the other way around. To this end we found metal fittings to secure the valves and then mounted the motors with a flexible fitting so that it would at first follow the non-stable valves before moving them.

We are using the MKR motor carrier to prototype and its design as a base for our project, which allows us to use the Arduino library specific to this board, the MKRMotorCarrier library. It includes functions for setting the angle of servos as well as the duty cycle for motors. The software will have hard coded values for the valve positions since these are something that cannot change without serious design changes. Each motor will be given a different value that signifies whether the valve will be turned to allow reagent to flow, inert gas to flow, or nothing to flow. These values will be saved in a separate header file with sufficient warnings to deter accidental manipulation.

The use of PWM signals to the motors controlling the valves will be done during strategic times within the program. The valves, which are three way valves, will be signaled to turn to a position that lets inert gas clear the lines first. After a set period of time, the valves will then be signaled to turn to allow reagents to flow. During an emergency shutdown or after a process has been completed, the valves will be signaled to turn to a neutral position that does not allow any material to pass through them.

C.LCD Display Menu

The LCD screen serves as the visual means of communication between the user and the hardware. It informs the user of current processes underway, display pertinent sensor data and any less critical warnings. As such, the screen needs to be large enough to display all the necessary text, though it does not have to display everything at the same time. In fact, it would be a better idea to keep the display as simple and uncluttered as possible to make the system more user-friendly. The display works in conjunction with a number of push buttons to allow the user to make choices which are then translated to the hardware.

The menu system needed to start at a place that gives the user the ability to start the current program, a quick start approach that will save time if the user is not changing between products after every batch. While a chemical process is being run, the display will show an estimated time to completion. This screen will be overwritten if there is a warning to display and the warning will not disappear until a user has acknowledged it via a push button. Warning that are displayed are not for issues that could be critical concerns, but for issues that can be addressed after the current process has finished. There are no options to stop a process while it is running at this time because there is a dedicated emergency shutdown system in place that would do much the same thing. The following Image shows the flow for the User interface.

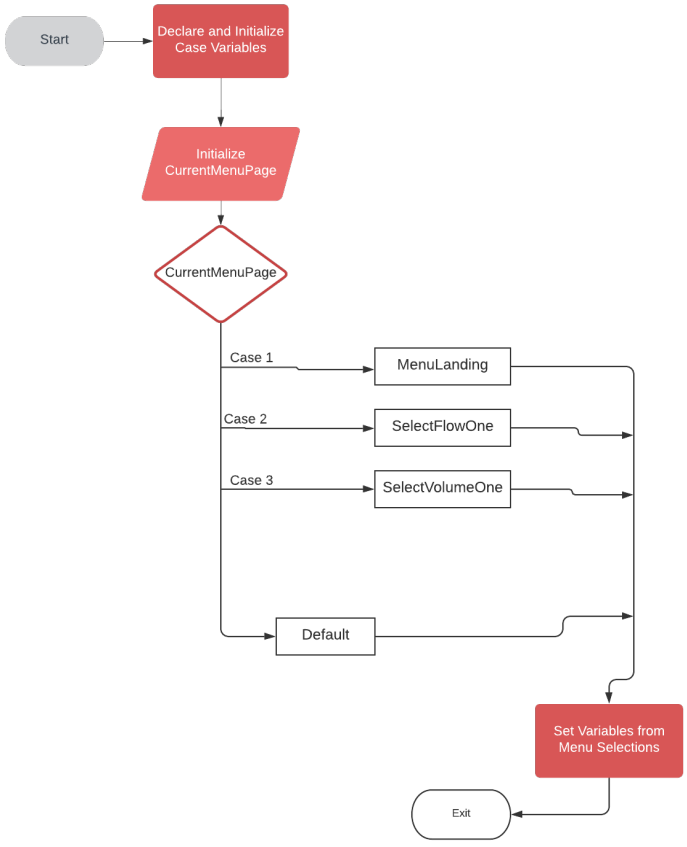


Figure 3 User interface flow chart.

VI. Conclusion

Although not perfect, our project was able to meet all the requirements we set out to achieve. This includes a PCB designed by our team, valves moving as instructed by the microcontroller, pumps interfaced by the

microcontroller, and a UI to tie everything together and allow the worker at helicon to control the system. Some improvements could have been made to the system and our approach to the problem but unforeseen circumstances also led to feasibility issues especially that of stretch goals.

ACKNOWLEDGEMENT

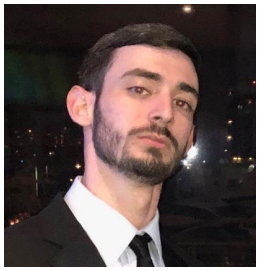
The authors wish to acknowledge the assistance and support of Carlos Reina who helped guide the team to solutions involving the enclosure and components, Ian McClure who was the head engineer building the system Helicon had and explained how exactly their current system operates and provided technical material, and David Reid, the Helicon CEO who authorized the project.

BIOGRAPHY

Amanda Gilliam is a senior Computer Engineering student at the University of Central Florida. Upon graduation she will join Northrop Grumman as a Software Engineer. Her interests include embedded development and software design.

Anish Umashankar is a Senior at UCF pursuing a bachelor's degree in Electrical Engineering. The choice to pursue Electrical Engineering as a major comes from being a child growing up in the digital age, which resulted in affinity towards electronics and circuit design. Personal hobbies include building computers for commission and making music. In the near future he aims to work in a job which he has appropriate creative freedom in PCB design.

Jason earned a business degree from the University of Florida and joined the Peace Corps before realizing that engineering was his true calling. He then came to UCF to study electrical engineering, with the ultimate goal of helping humanity colonize other planets. In his free time he enjoys learning, teaching, and traveling. The ultimate combination of his goals and interests is to one day travel to Mars.



Ernel is a senior Electrical Engineering student at the University of Central Florida. Electricity is the family trade and Ernel grew to enjoy automation after helping his father automate an Ice plant. He aims to secure a job in Siemens or FPL doing automation work.

REFERENCES

[1] “IEEE Standard for Software Test Documentation," in IEEE Std 829-1983 , vol., no., pp.1-48, 18 Feb. 1983.

[2] “NEMA Enclosure Types” in NEMA Enclosure Types vol., no., pp.1-9, Nov. 2005.