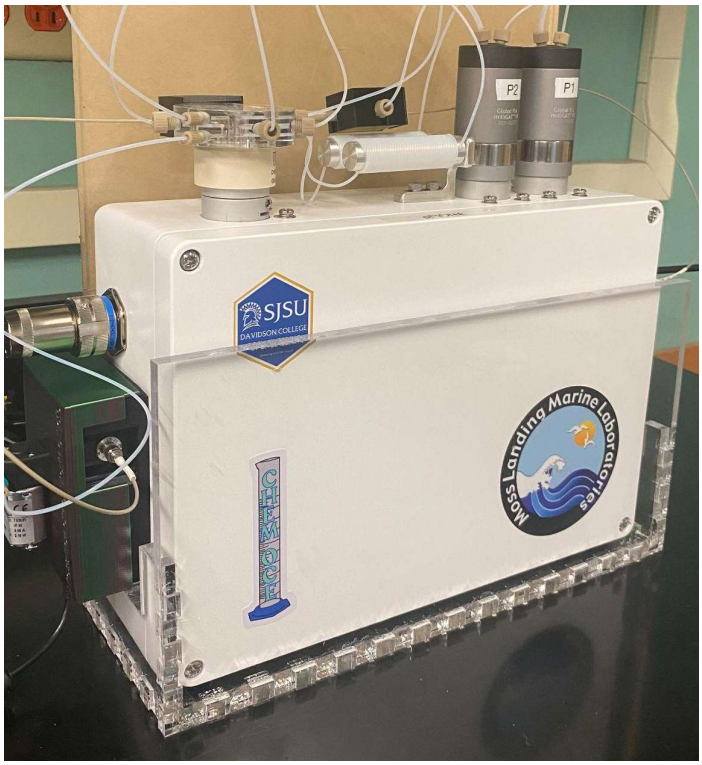
SAN JOSÉ STATE UNIVERSITY | CHARLES W. DAVIDSON COLLEGE OF ENGINEERING

Senior Design Project

Project Report:

Programmable Flow Injection Ocean Nutrient Analyzer (pFIONA)



ME 195A Section 06

Instructor: Edward Cydzik

Authors: A. Silva, T. Hunter, T. Ly

Department of Mechanical Engineering, San José State University

May 16, 2022

Andrew Silva (Team Lead)

Responsible for research and development of firmware. Hardware selection, enclosure design, and assembly. Component drawings, circuit board design, and fabrication.

Timothy Ly

Responsible for implementation of spectrometer, data collection, and processing. Software debugging and assistance.

Tracy Hunter

Responsible for additive manufacturing of mounting fixtures, implementing code for Chem-On Valve servo, MilliGat pumps control, and end-user variables for ease of system control.

Abstract

The purpose of this report is to document the process of designing and developing an automated programmable flow injection analysis (pFIA) instrument for field use. This instrument was reverse engineered from a benchtop model that was designed to be operated by a user with a personal computer (PC) using proprietary software sold by the manufacturer of the instrument. Research into firmware and the requisite hardware is discussed as the primary challenge of the project. Iterative design methodologies allowed for rapid product development after initial establishment of communication protocols. The project used a Rasberry Pi to mimic the PC's serial connection to the proprietary (undocumented) communication board, which served as the interface and power distribution printed circuit board (PCB) for all devices used in the instrument.

Acknowledgements

The team would like to express their gratitude to the team from Moss Landing Marine Lab, Maxime Grand, Marine Lebrec, Haonan Wang, Aakanksha Gupta, and the team of advisors from San José State University, Crystal Han, Farzan Kazimifar, and Ed Cydzik. Their effort and support to help guide us on the project and keep us on track to create a functioning analyzer proved to be invaluable. They allowed us to gain knowledge in project management, system implementation, design, and learning to work with a multidisciplinary team. Additionally we would like to thank Dr. Winncy Du and Dr. Lin Jiang for allowing us to utilize the robotics lab to work on the project for the year. Lastly, we would also like to extend our appreciation to Dr. Furman and Eric Wertz for helping us by answering technical questions in order for us to complete the project.

Table of Contents

Executive Summary

In order to measure phosphate (PO4) levels in the sea water of Moss Landing, the Moss Landing Marine Laboratory (MLML) purchases analyzer machines whose base price is $35,000 plus an additional $1,200 software to operate them. In order to cut costs, the Laboratory created a project to build a microcomputer based machine that integrates the same components that, according to the lab, can save tens of thousands of dollars while providing a higher resolution of phosphate samples.

The core components were provided for the project by the MLML with the exception of the microcomputer. This included two aspirating/dispensing step motors, a rotary valve connected to a motor, a communication board, spectrophotometer, and a power supply. The first task was to choose a suitable microcomputer that had sufficient speed , memory, and storage. The Raspberry Pi 4 was chosen as it provided exceptional performance in all three criteria.

The next step was to establish communication between the microcomputer and the hardware. The hardware provided included a communication board that handled power delivery and communication between the components and a chipset. It was known that the chip could be interfaced via usb connection using a windows computer. In order to simulate the connection, the pySerial[18] library was downloaded on the Raspberry Pi. This established a serial object that could be used to send encoded strings to the chipset that would handle communication to the motors.

The motors feature a controller integrated into them, and can operate the motor via proprietary commands. By using the communication board with the serial object in the project program, it is possible to encode the proprietary commands in the python script and send it to the communication board, which then sends the command to the motor controller and operates the motor. This is the main method for controlling both the valve and the motor of the project. The script was then designed based on the sequence provided by the laboratory. The motors and valves successfully changed ports and dispensed/aspirated the proper amounts.

The final step in recreating the commercial flow analyzer is handling the data collection and processing. The spectrophotometer interfaces directly with the Raspberry Pi and can be controlled through the Seabreeze library[17] for Python. The library enables detection of the

spectrophotometer via USB connection and access to the data collected by the device. By programming a LED bulb to operate off of the pins on the Raspberry Pi, light can be passed through an optic fiber, through a flow cell filled with the chemical sample, and finally to the spectrophotometer that will measure the intensity of the light. Using this data, the absorbance of the sample can be collected and the levels of phosphate in the sample can be measured, simulating the results of the commercial machine, although without a graphical user interface.

Chapter 1

Problem Definition

In the waters of Monterey bay in California exists a vast abundance of oceanic diversity. These nutrient rich waters contain an ecosystem which includes 33 species of marine animals, 94 species of seabirds, as well as 345 species of fish [1]. To maintain this ecosystem one needs to better understand the underlying parameters that impact it. Two of these parameters are nitrogen and phosphorus levels in the ocean. These two elements are essential for plant life, however when there is an abundance of dissolved nutrients in water it can produce the phenomenon known as eutrophication, which results in the reduction of dissolved oxygen in water [2]. This influx of nutrients produces massive algae growth, which is detrimental to the ecosystem. Different strains of algae produce different byproducts as a result of massive growth, the most devastating effects can be red tides, oxygen depletion, poisonous toxins, as well as fish gill damage [3]. One main culprit of this influx of phosphorus and nitrate levels is caused by the use of fertilizers for large agriculture businesses. The fertilizers used contain nitrates and phosphates which help crops grow but this overuse can reach into the waterways and throw off the balance of whole ecosystems. Therefore a way to monitor the compounds is essential. Monterey Bay is home not only to a biodiverse ecosystem, it is also a vast agricultural hub in the U.S. Monterey county feeds the nation by supplying 61% of leaf lettuce, 57% of celery, 56% head lettuce, 48% of broccoli, 38% spinach, 30% cauliflower, 28% strawberries, and 3.6% of wines grapes across 267,873 acres of land [4]. Therefore there is a need for monitoring the levels of nitrate and phosphate closely.

Nutrient analyzers are instruments which are capable of measuring concentrations of certain nutrients. Most measurements of nutrients are done by taking water samples for later analysis at the lab. One method of a nutrient analyzer is using a wet chemical process. The process draws in sample water on the device which is mixed with reagents, resulting in a color complex. Using this complex in conjunction with a light source, one can use a spectrometer to measure the concentration of target nutrients [5]. Many nutrients can be measured such as dissolved nitrate, ammonium phosphate and silicate.

Deployable instruments are expensive, in the tens of thousands of dollars, and units in the field are deployed sparsely. The nutrient analyzer project aims to lower this price by not commercializing it. Instead, it will be made available to everyone who wants to build it. This will provide increased monitoring of water supplies. This greatly increases the equality factor because it provides everyone access to water monitoring, not just those that can afford it. Socially it can impact people's ways of life by ensuring water quality standards. Many big agricultural businesses use fertilizers and pesticides which contain high levels of phosphates, cattle farming has manure which also contain high levels of nitrogen. When these are not monitored, it destroys marine wildlife and has dire consequences for humans as well. The normal phosphorus level is 2.5 to 4.5mg/dl, however when these are elevated, they can create changes in the body that pull calcium from the bones which leads to dangerous calcium deposits in blood vessels, lungs, eyes, and heart which can lead to increase chance of heart attack and stroke [6]. Having monitoring sites that are farther upstream to the source of pollution, the instrument can help pinpoint the source of pollution and create more accountability to those offenders of the laws that are put in place. Unlike more developed areas like larger cities which have large treatment facilities, areas surrounding big Ag and industry are more of the lower class farm workers which rely on underground water aquifers for their clean water.

Project Objectives and Specification

The primary objective of this project is to design and implement an autonomous version of the current benchtop nutrient analyzer that is in use by the Moss Landing Marine Laboratory. This Primary objective will be achieved by implementing a microcomputer into the current design to

bypass the communications board and software that is used. The draw back now is that a desktop or laptop computer must be used in order to run the system and analyze data. This makes for a restriction in the deployable process. Currently samples are taken from the field and brought back to the lab for analysis which is time consuming. Furthermore, samples are just instances at specific times and do not provide real time data collection capabilities. By implementing a microcomputer, data will be collected from the field in real time to monitor water quality in terms of phosphate levels. These time series will then be stored on the drive of the controller where a researcher can access it periodically. A sub-objective of this project, if time permits, will be to design a storage container to house the two pumps, the servo valve, spectrometer, reagents, waste bag, and power supply unit.

Design Specifications

This project was focused on the automation of a (pFIA) instrument to be used to monitor water quality at the Moss Landing Marine Lab. The design specifications will be to take the currently working desktop version of the analyzer and create a self-contained one. Meaning a device that takes samples in-situ without the need of a user and their PC. The instrument is to be in service in a pump house of the Monterey Bay Aquatic Research Institute (MBARI), a noisy and wet industrial setting Just off the coast of the Monterey Bay. So the device needed to be able to function in an industrially noisey, wet and corrosive environment. Special considerations had to be made to accommodate the protection of the vital equipment balanced with access to the serviceable components.

Current Status (Literature Review)

Nutrient analyzers are used in abundance throughout many fields of industry. For example, they are installed in wastewater treatment plants to ensure regulatory compliance by controlling the outlets of wastewater inside of the facility. By monitoring the nutrient levels they help to optimize the aeration control and precipitant dosing during biological treatment of wastewater, and monitor denitrification to support safe drinking water, mineral water and process water [7]. Liquid analysis

is essential in many industries to achieve high quality and efficient process control. One of the most important aspects to analysis is the colormetric measuring principle and photometric principle, which is employed in the current version used by the Moss Landing Marine Lab (MLML). The instrument in use is GlobalFIAs miniSIA-2 [8].

Figure 1

The miniSIA-2 analyzer with Chem-on-Valve manifold courtesy www.globalfia.com

From researching GlobalFIA's website we learned that this system is equipped with two milliGAT pumps coupled with heating coils, which control the speed of the reagent/sample mixing. Tubes running from these coils meet at the center of the Chem-onValve (VOC). The valve and pumps are controlled through the use of an Mdrive-17 plus microstepper motor, which allows for precision control through its high resolution.One of the main advantages of this type of setup is that it allows for simultaneous flow control with bi-directional movement of fluid between the pumps. This coupled with the multiport valve allows for an enclosed movement of reagents and samples which reduces the risk of outside contamination like when assaying on a wet bench. However, there is one drawback to this system in that if the pumps are not under precise control,

one can easily contaminate the sample reservoir. If one pump stops aspirating before the second which is dispensing, it can create reverse flow leading to the sample line.

Through a sequence of steps this instrument mixes reagents and samples which are then moved through the system to the Spectrometer for analysis. MLML currently uses this process developed by GlobalFIA to analyze samples. While their company uses its own programming through their Flo-ZF software, it is a robust application that can be set to the needs of the consumer. One is allowed to create their own set of commands and controls for the minSIA-2 to suit their needs. The researchers at MLML have spent the last year and a half creating their own chemistry to detect phosphate concentrations in their sample collected. One of the assistant professors at MLML, Maxime Grand, and his colleagues created their own chemical assay to detect phosphate concentrations with repeatable results.

Relevant Research

It was important to conduct thorough research into the driving principles behind the project goals. Everyone on the team has a background in engineering with only a cursory level of understanding of chemistry. This project is deeply entrenched in principles in organic chemistry, and more specifically, organic ocean chemistry. The 24-step assay used for capturing data required us to develop a conceptual background in this field to better assess the problem. We wanted to see what solutions others had made in the field and a focus on previously proven methods of nutrient detection and quantification was enacted; resulting in an abundance of information. A method of detection known as the molybdenum blue method as described by N. Ibnul et. al. details "the American Public Health Association (APHA) approved method for the detection and quantification of phosphate in water. The standard molybdenum blue method, APHA 4500 PE has a detection limit of 30 μgL−1 phosphate (10 μgL−1 phosphorus) in freshwater with a 5 cm cuvette," [9]. This study focuses on freshwater phosphate samples, which is the exact same chemical assay used for seawater, in that fresh and sea water produce identical results regardless of the two mediums. The overall process and the emphasis on following a method in accordance with the APHA was determined to be important. The project is grounded in sustainability and it's paramount that the team's solution reflects that. Defining result expectations were salient for the

research as well. We can see from the presented data that we can expect a resolution as low as 4umol/kg for phosphate anions and a resolution as high as 50nmol/kg for phosphorus. We used this to characterize the experimental expectations for the max limit of accuracy of data values we will capture.

One of the project's goals is the ability to massively increase the rate of data collection. Right now, the current rate of data capture for the bench top setup is one sample every 3.5 min. However, the more samples taken, the more work will be required for collecting samples in the field to bring back to the lab. Legiret et. al. presents an experimental set that was capable of creating a "measurement frequency [that] was configurable with a sampling throughput of up to 20 samples per hour," [10]. The set up was also "[a] portable micro-analytical system [which] has a low reagent requirement (340 μL per sample) and power consumption (756 J per sample), and has allowed accurate high resolution measurements of soluble reactive phosphorus in seawater," [10]. A portable solution that will be deployed for extended periods of time is going to require energy management, reagent management, and data capture protocol. This research demonstrates that it is possible for seawater applications and further characterizes energy expectations per sample and reagent used per sample. These values are key to the initial design, allowing us to approximate what type of requirements we are going to need to meet. The goal was to measure one sample every hour for 30 days. Where Legiret et. al showcases a high frequency data capture solution, we are looking for a more robust, long term setup. While the above research indicates that sampling as fast as 20 samples per hour is possible, it is not detailed how long this module lasts in the field.

The research presented above shows solutions to problems very similar to our own. Characterizing how fine of a measurement we can make, how often, energy consumption per sample and reagent used per sample enables us to fill in the blanks of some of our design parameters. This research gave us confidence that a portable nutrient analyzer is feasible within the scope of our project and aided in the advancement of our design.

Significance of Project and Challenges

This project offers multiple avenues of significance as well as its own unique set of challenges. Developing a deployable nutrient analysis within the framework of open source technology offers

wide access to the scientific community. Water quality is imperative and commercial options often exceed what research is capable of funding. We are removing barriers to entry for measuring environmental issues. With the emphasis on our project being able to measure agricultural run off we are working to provide technology that combats environmental hazards and promotes sustainability. All of us convergently seeked this project out because of a love of nature and maintaining balance within our natural world. As engineers we have a significant impact over how the world advances and with the right effort we can provide alternatives to the status quo that promotes sustainability and democratizing water quality analysis.

This project came with unique challenges. Interfacing with a less familiar science discipline to automate a chemical process provided communication challenges. Working within the constraints of a pandemic bottlenecked our access to parts and manufacturing capabilities. We combatted this by reallocating time and effort to areas where we can make an impact on the project. The issue of integrating commercial proprietary (undocumented) equipment with open source development had proven to be a challenge.

Team Work

Each of us were tasked with specific sections of the project ranging from pure programming to bill of material analysis. Each of us worked on everything but had focus. Tracy was responsible for implementation of the python script for controlling the servo valve and pumps. Additionally work on implementing end-user robustness for ease of control changes to parameters with minimal coding knowledge. Additive manufacturing for components was also tackled as well as conducting a literature review. Andrew has been tackling the majority of communication protocol troubleshooting. This is being conducted through conferring with documentation and actual analysis of the system in person. Timothy has undertaken the majority of the control system design study as well as programming methodology study.

Specifications

The MLML has requested the following specification of the instrument. Primarily that it can run autonomously and continuously, for as long as it has enough substrate. This requires multiple

feedback loops to ensure the instrument does not perform incorrectly and contaminate the analysis. Secondarily, the instrument needs to be able to perform the calculations on its own and be able to record the findings. The instrument will run its own calibration process to ensure data collected is accurate. The data logged is an absorbance value calculated from intensities and wavelengths obtained from the spectrophotometer.

Chapter 2

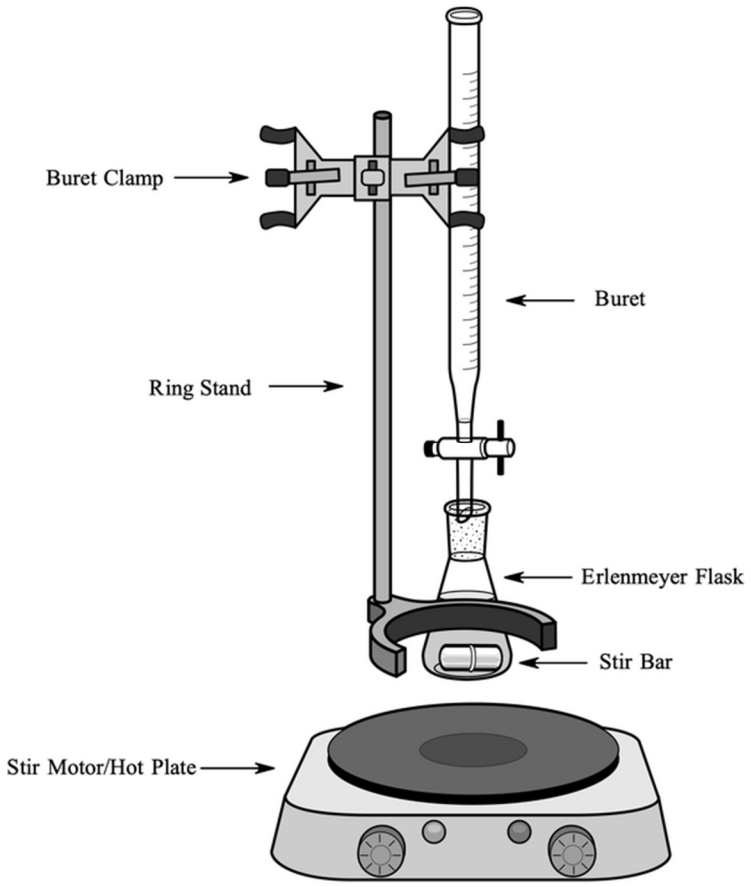
Theoretical Background

Solution handling is one of the widely applied laboratory techniques. In analytical labs it tends to be one of the most laborious tasks due to it needing to be monitored constantly in order to be precisely mixed, incubated, separated, etc. [11]. Still today there are many chemical assays that continue to be carried out manually and have remained unchanged in the last 200 years. Recently this process has been down-scaled to an automation approach which eliminates the need for batch-sample processing and introduces the concept of flow control processing [11]. The flow control process of the minSIA-2 is made simple by having very few mechanical parts, in this sense it only has a pump for dispensing and aspirating and a valve for sample/reagent control.

Reagent based assays rely on homogenous mixing techniques. One of the first quantitative analysis techniques was that of gravimetry, in which the mass of an ion in a pure compound can be determined [12]. Later Antoine-Laurent de Lavoisier, a French nobleman and chemist, developed titrimetry, which is a quantitation technique involving the absolute method based on stoichiometric reactions to accurately determine the concentration of an analyte based on volumetric scaling [13]. Basically the equilibrium between the analyte and titrant was achieved through a stepwise process until the equivalence was achieved (fig.2). This led to the principles of homogenous mixing and reaction equilibrium which still to this day are practiced and applied to reagent-based lab techniques.

Figure 2

The basic setup of the tetimetry analysis on a wet bench. Photo courtesy of www.qph.fs.quoracdn.ne



Recently over the past 50 years the reaction rate measurement and enzymatic assays principles have become the norm. The reaction rate can be defined as the change in concentration divided by the change in time [14]. This concept allowed for those chemical reaction based assays not to reach full equilibrium to make analytical conclusions. Furthermore, the introduction to flow based techniques are a completely different concept altogether. These flow based techniques is what leads us to the main concept behind the GlobalFIAs miniSIA-2 platform, the flow injection analysis.

Flow injection analysis (FIA) is a fluid analysis method involving sample injection, transport, reagent addition, reaction, and detection that can be done quickly (up to 10s of seconds quick) while remaining efficient with resources and a high level of precision [14]. Its efficiency of resources derives from FIA's use of small tubing as small as .3 mm to promote a smooth laminar flow. Proportionally, this results in small amounts of reagents and reactants to be used in order to complete the necessary chemical reactions, increasing speed as well.

FIA involves a propulsion, injection, separation+mixing, and a detection component. Propulsion components usually consist of a pump or syringe for delivery of carrier and reagent solutions. A popular choice in early FIA development was the peristaltic pump due to its availability and multiple channels. However, the peristaltic pumps actuates using pulses, and thus as the materials wear the flow produced with the pump will reduce over time. More recently developed, the milliGAT pump, used in our project, consists of four, spring loaded reciprocating piston driven by a stepper motor. As one piston on the motor fills a port in the cylinder head of the pump, a second piston discharges through a second port. This results in an overall pulseless flow out of the pump, solving the problems of the peristaltic pump. The pump is also reversible in direction allowing it to both aspirate and dispense. The milliGAT pump delivers flows from nanoliters/min to milliliters/min [14]. The pump works to move fluid throughout the system, using both aspiration and dispensing to perform the necessary reactions by moving the sample or chemical into the proper mixing chamber.

The injection component of our study consists of a rotary valve. Early models used a septum and syringe to inject samples into the system. The need for high-precision and automation led to the development of the rotary valve which is driven by stepper motors The rotary valve utilizes time based injection, which is pumping through a manifold at a known flow rate for a defined period. This allows injection volume to be varied by altering the time a valve is open. This allows different fluids to be used as timing would just need to be changed to accommodate for different mechanical properties. However, any external disruptions to flow rate will greatly affect the reproducibility of the process, meaning error handling will be necessary [14]. This component can control the piping system of the process and can connect necessary valves and containers, switching between samples and necessary chemicals for the process.

The separation+mixing component of our study consists of a PTFE coiled tubing. PTFE is used because of its chemically unreactive property. Due to the small internal diameter of the tubing, laminar flow is maintained throughout the system. The coiled portion prompts radial mixing of the reagents and reactants as the fluids flow in a radial motion through the coil. If a small enough coil radius is used, secondary flow is induced and promotes the radial mixing. Using two coils, it is possible to mix between them as well as transport fluids from one to the other. The coils act as a zone for chemicals to mix and keep them separated from the rest of the system until it is necessary to proceed with the process [15].

The detection component uses a sensor for measurement of change in a variety of properties including fluorescence, infrared absorption, pH, or, in our case, absorbance. The project uses a spectrophotometer, a device that measures light transmittance through a solution by passing a light through and measuring light intensity as a function of wavelength [16]. This requires a base reading for reference with deionized water and the absence of air bubbles that can reflect the light shining into the substance. The FIA system, after completing the mixture of chemicals and sample, moves the mixture into a tube where light is shining on one end, through the tube, and into the spectrophotometer. The spectrophotometer records the data measured and from it the concentration of a target substance can be determined.

FIA combines all these components to develop a way to perform fluid analysis efficiently and quickly. From the precision of the pump, small and precise amounts of chemicals and samples can be used resulting in faster reaction times while maintaining accuracy. The injection component regulates the flow of the fluids throughout the system into the separation, mixing, reactants, and reagent components. Finally, the detection component takes the measurements necessary for analysis of the concentration of the target substance.

Data Processing and Calculation

The spectrometer is able to take values for light intensities across wavelengths 200-850nm. The absorbance is calculated as per the following equation:

$$A = log_{10}\frac{I_o}{I} \quad (1)$$

$$A = -log_{10}T \quad (2)$$

$$I_o = Incident\ Intensity$$

$$I = Transmitted\ Intensity$$

$$T = Transmittance = \frac{I}{I_o}$$

$$A = Absorbance$$

The Incident Intensity is the reading from the light,which is taken with pure deionized water in the tube , subtracted by a dark reading (with deionized water in the tube) to zero the error when comparing with the transmitted intensity. The Transmitted Intensity is the reading with the chemically mixed sample in the tube, subtracted by the dark reading.

Once the absorbance is calculated, it is then averaged and filtered to values less than a monitoring wavelength and reference wavelength. The final absorbance is taken from the difference between the two averaged filtered absorbances.

By taking the final absorbances from a blank sample (deionized water+reagents) run, PO4 standard run, and the sample run, the concentration of PO4 is calculated by the following equation:

$$PO_4 = (S_{abs} - MB_{abs}) * (\frac{KCPO_4}{MS_{abs} - MB_{abs}}) \quad (3)$$

$$S_{abs} = Sample\ Absorbance$$

$$MB_{abs} = Mean\ Value\ Blank\ Absorbance$$

$$KCPO_4 = Known\ Concentration\ PO_4\ Standard$$

$$MS_{abs} = Mean\ Value\ Standard\ Absorbance$$

Where these values are:

$$MB_{abs} = \frac{Blank\ Absorbance\ Values}{Blank\ Absorbance\ Runs}$$

$$MS_{abs} = \frac{Standard\ Absorbance\ Values}{Standard\ Absorbance\ Runs}$$

Chapter 3

Design Overview

Much of the design phase had occurred prior to the start of the team drafting this report. The main components and supporting hardware for the instrument were provided by the project's sponsor the MLML. It was initially theorized that the functionality of the components provided was such that the existing communication PCB did not benefit the autonomization of the instrument. It was discovered during research that this communication PCB communicates with the pumps and servo using the RS485 standard. This PCB uses the FT232R chip from FTDI to convert the serial connection from the user's PC to the RS485 bus which the three Mdrive 17 pluses operate on. TUSB2077a on the PCB creates virtual usb ports that are addresses for each device on the bus, also serves as the memory for the device registry referenced in the programming as A, C, and D. The MAX485 IC on the communication PCB serves as the transceiver for the RS485 bus. Gaining this understanding and developing the requisite firmware proved to be the primary challenge of this project.

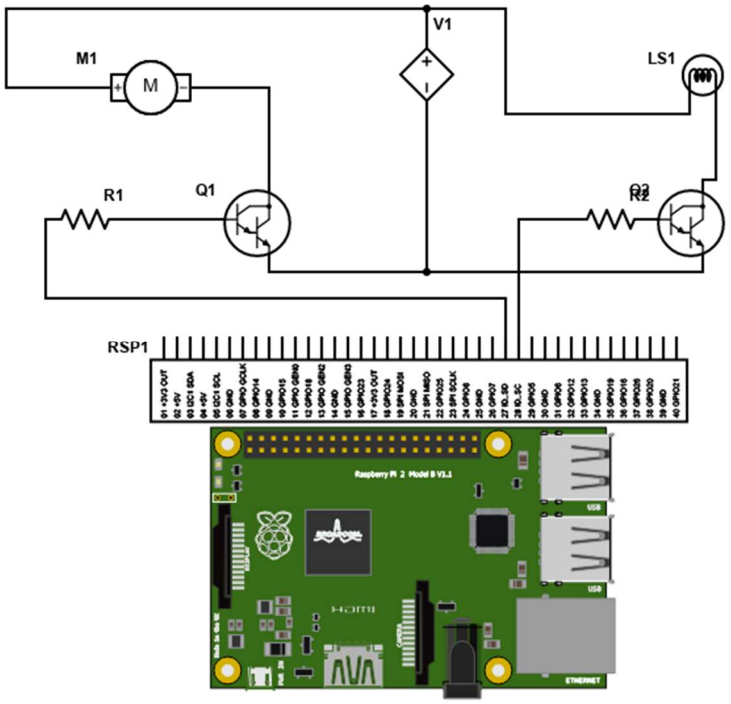
In order to overcome these challenges, the advice of a diverse group of advisors proved to yield a method of control for the devices. The communication PCB was designed to connect to a user's PC in a way a microcomputer can emulate as long as it has host USB capabilities. The project's initial design did not incorporate the communication PCB used in the previous instrument. This reduction of parts would have reduced the overall cost of the instrument by an estimated $400, as well as lowers the barriers to entry of building an pFIA device. After further research and due to delays in part delivery, time and supply constraints, the use of this PCB was integrated into the design and research into its replacement was allocated to the future works section. The use of this

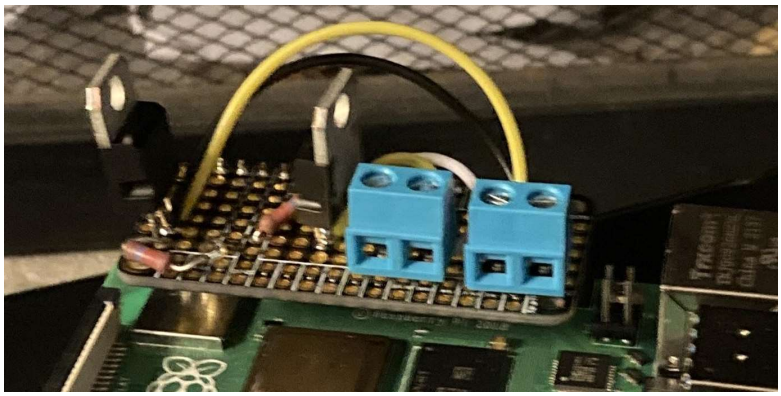
communication board simplified connections, as custom circuits were no longer required to interface the microcomputer to the pump and valve motors, nor to supply power to all the devices.

The simplicity of working with an integrated system provided by using the GlobalFIA hardware helped to facilitate the rapid development of the instrument. The power delivery and control are handled by the communication board as it interfaces with a power supply to deliver power to all the components of the system. The final hardware included two Mdrives with Miligat high flow pumps, one Mdrive with Chem-on-Valve selector, 120 watt power supply, 20 watt incandescent, auxiliary pump, Raspberry Pi 4, spectrophotometer, two optic fibers, flow cell, tubing, and two holding coils. The Raspberry Pi 4 controls the motors through the serial connection to the communication PCB and control the auxiliary pump, LED, and spectrophotometer through a custom circuit board see below in figure 3.

Figure 3

Circuit board diagram and actual circuit board for controlling the Auxiliary pump and light source with the Raspberry Pi's GPIO pins

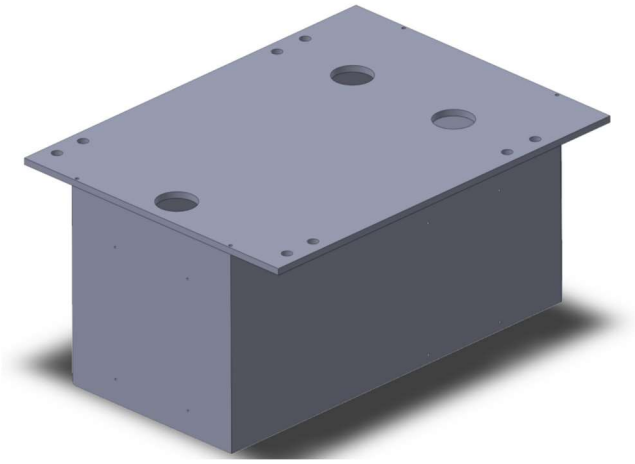




Enclosure Design

Figure 4

Initial proposed enclosure for instrument

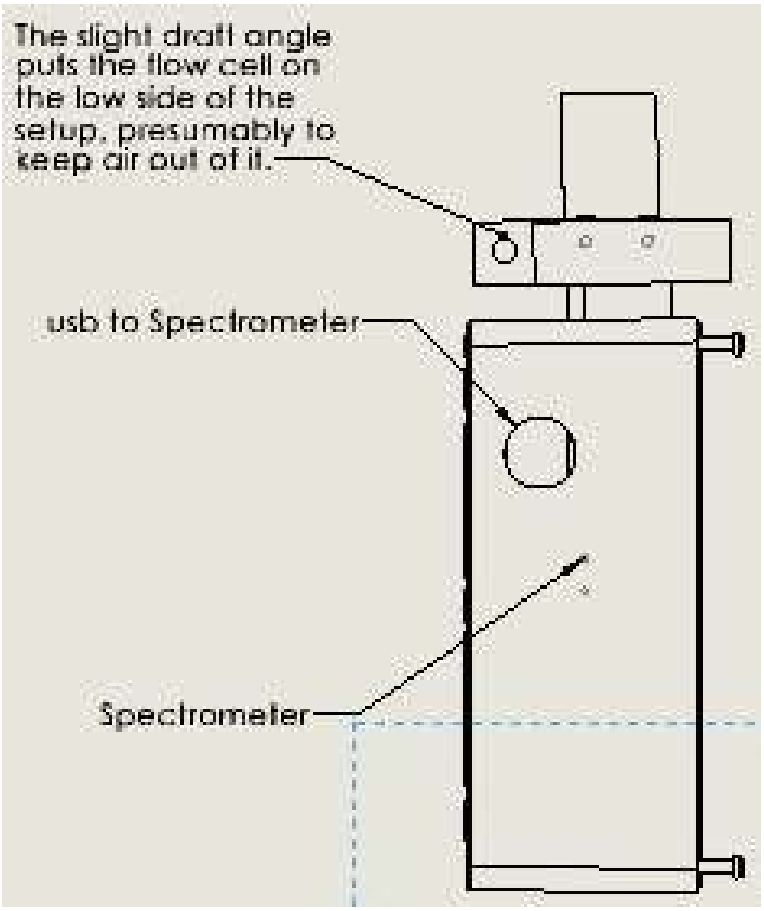


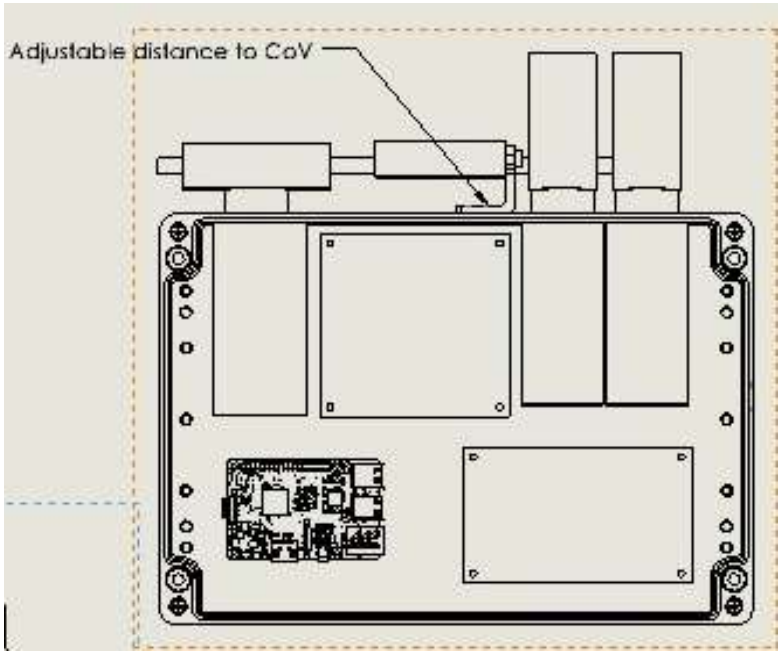
When considering enclosure designs it is important to consider the environment the instrument will be exposed to. The MLML uses this instrument in a coastal environment where salt water is prevalent and specific considerations need to be made. In this environment galvanic corrosion is prevalent, the high moisture content contributes to this as well as shortening the lifespan of electronics exposed to it. In figure (4) above the initial design is shown for the primary enclosure.

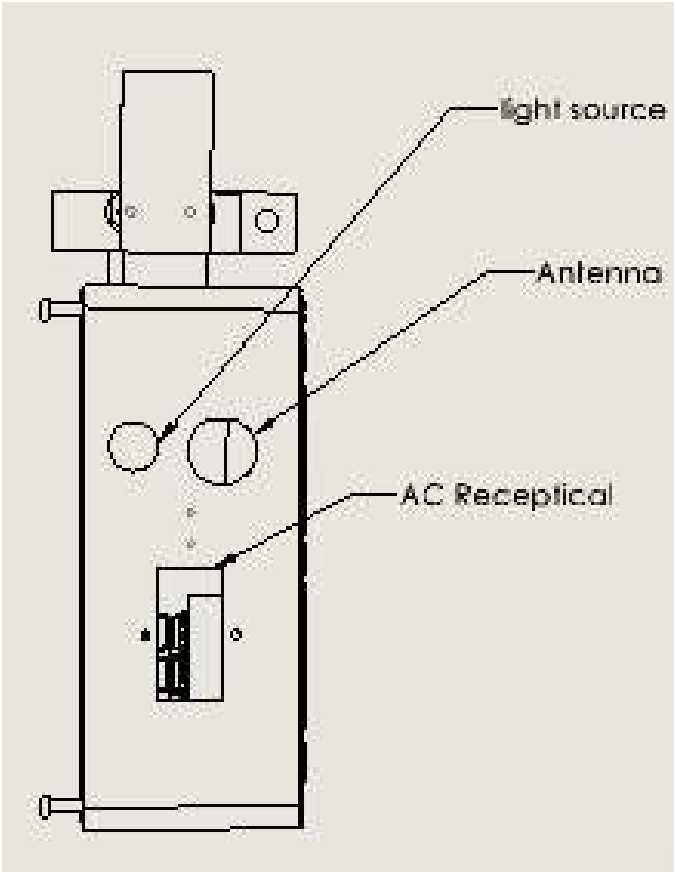
This design mimicked the one used by GlobalFIA. It would have required more tooling and production expense than the final solution of the modified commercially available submersible electrical enclosure.

Figure 5

Notable features of the enclosures design



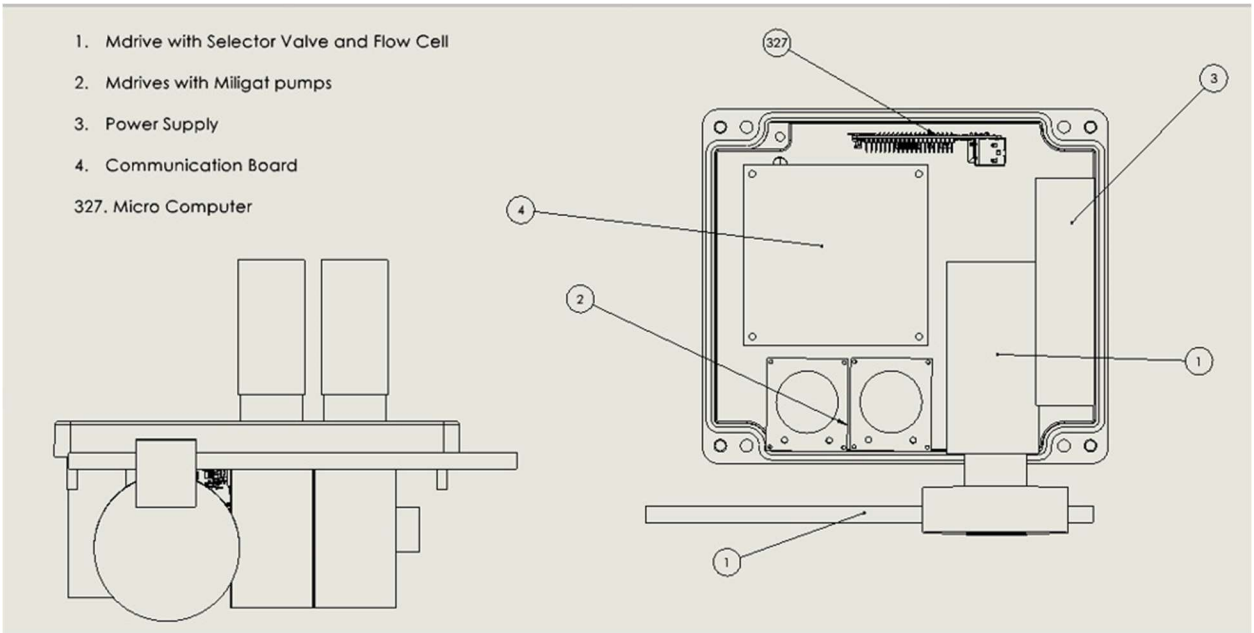

Adjustable distance to CoV


light source
Antenna
AC Receptical

In the following image the figure (6) housing design for the pumps and servos can be seen inside the enclosure. This design enclosed all of the primary devices, control and data collection device inside a sealed aluminum enclosure. Special care was given to select watertight pass-throughs for the two Miligat pumps, the CoV Selector Valve, the lightsource, the WiFi Antenna, the auxiliary pump, and the spectrometer. Additional consideration was given to the potentially electrically noisy environment the instrument will be operating in which resulted in USB data cable with Ferro shields. A semi-drying sealant was used on all mating surfaces of components to ensure the case remains water tight. Rudimentary analysis into the amount of heat dissipated by the device's painted aluminum surfaces proved to be adequate passive cooling for the devices maximum 120 watts.

Figure 6

This image shows the third model created to decide orientation of the components. The pumps in this version go through the lid, creating increased difficulties in servicing the Instrument.

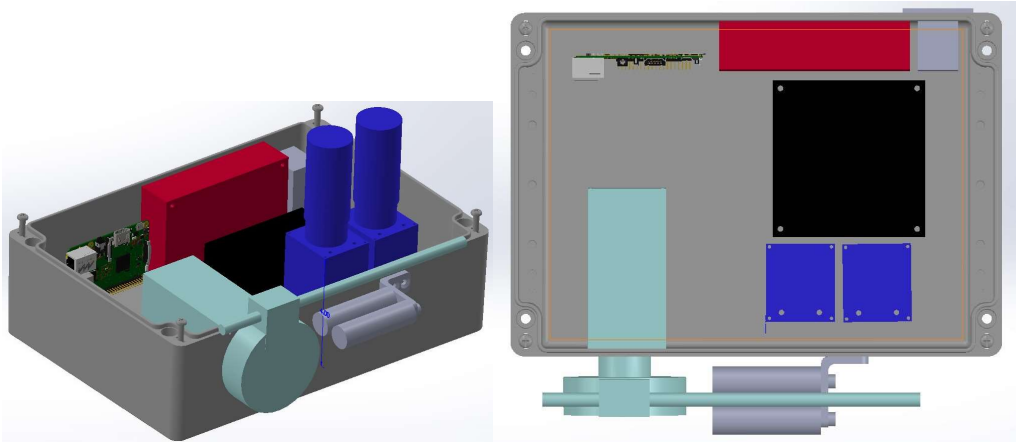


In order to facilitate the needs of the chemistry being automated, iterative design techniques were utilized. For the enclosure and orientation of the components three dimensional models of each component were created to be used in an assembly model in order to facilitate the back and forth

with the chemist end users. The figure below shows the initial orientation of components using the larger aluminum enclosure, the second to last enclosure designs showcasing one of the many three dimensional assembly models created using SolidWorks. This method of building digital representations allowed for maximizing the time we had to communicate with the team as the MLML as well as to quickly and cost effectively change the revision of the enclosure's design.

Figure 7

These images show the component layout that was runner up in the enclosure design choices.



After confirmation of the suggested component orientation and enclosure layout the design took the form as shown in the following two figures. This orientation allowed for minimal footprint and maximum exposed surfaces for passive cooling. The 'on edge' orientation also allows for the enclosure's removable cover to be readily removed for access to internal componentry. Figure (8) below shows the components as they exist in the enclosure. It is a very snug fit with all of the components attached and the needed wiring installed. In order to prevent short circuits, stand offs were made using a 3D printer for the electronic components inside the enclosure. These parts are shown in the appendix under drawings.

The use of an external antenna was necessary due to the encasement of the MCU and its internal WiFi antenna. The enclosure includes a rubber gasket in a groove to seal the lid that is replaceable

from the supplier McMaster-Carr. Three dimensional models were created for each component and several orientations of assemblies of these components were iterated to interface with the team at the MLML and to better understand that the requirements of the assay were being met. Initially a smaller version of this enclosure was considered, but the in-ability to remove the need for the communication PCB required more space.

Figure 8

Components of the instrument as designed.

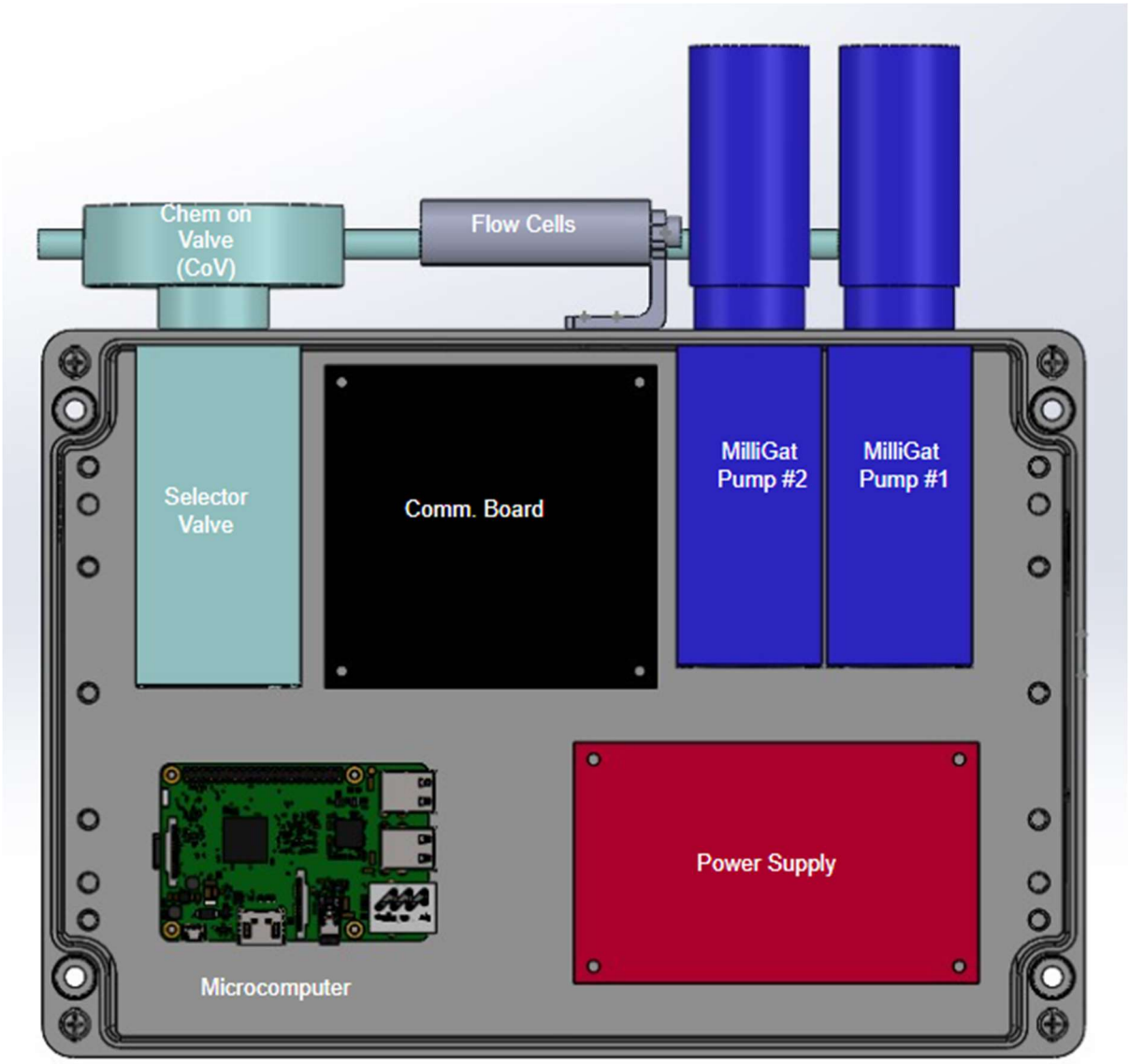
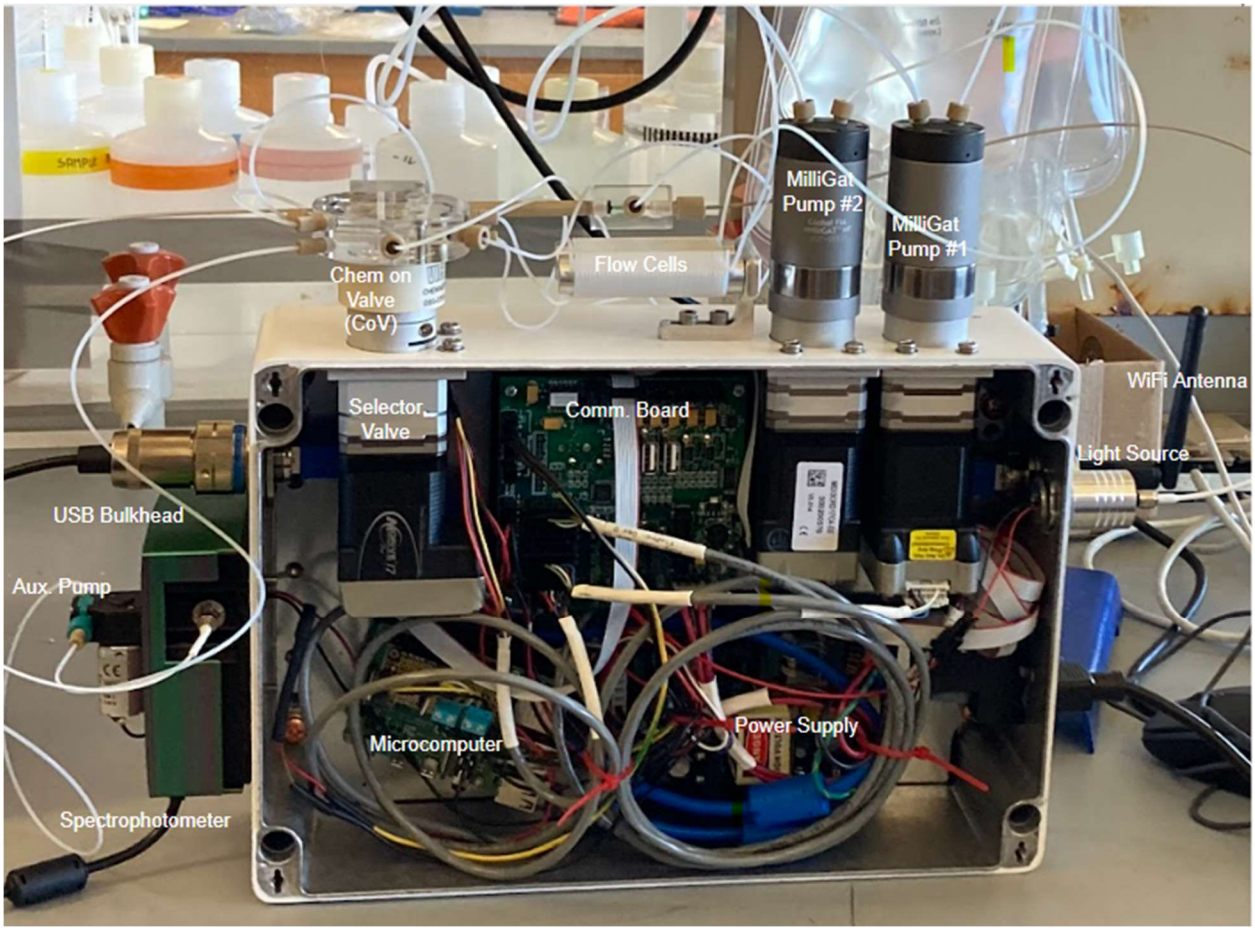


Figure 9

Components' orientations of the instrument as implemented.



Mechatronic System

At the center of this instrument's design is a microcomputer (MCU). This MCU will be managing the light source, Mdrive's, auxiliary pump and servo used in the assay. There are many microcomputers that fit the parameters of operation needed. The communication protocol required at least three USB 2.0 or higher ports with at least one having host capabilities. Initial development was done with an All Weather H3 chipset based MCU called an Orange Pi PC+. This development board fit the needs of the design and was readily available at a standard price long after supply shortages more than tripled the price of Raspberry Pi's. Both MCU's used a version of Debian, a linux based operating system.

Further research and expert consultation led us to decide on the Raspberry Pi 4 for its widespread availability, support and continued development. The latest version offered a quad core clocked at 1.5Ghz, 2GB ram, expandable storage, WiFi 5 for wireless communication, and 4 USB ports for connectivity between hardware. The team decided that for the first rendition of the instrument, considering the potential scope of data processing, to select a high end microcomputer such as this for faster debugging, runtime, and potential for high loads.

The microcomputer has expandable storage, and we found the ability to remove the SDcard to access the data or to be able to increase storage a beneficial feature. The MCU features WiFi 5, but the speed of wifi connection was not a large concern as wireless control did not have strict requirements for fast speeds. However, the benefits in range of WiFi 5 would be a positive addition for the project as potential debugging could be done wirelessly through either SSH Linux command prompt based interfacing or, a virtual desktop service such as VNC viewer client open source software. The MCU needed a minimum of 2 USB ports. One for the spectrophotometer and one for the communication board. As the Raspberry Pi 4 had 4 USB ports, two of which have host capabilities, it exceeded the project's requirements and allowed for future additions to the instrument.

Suggestions for such additions can be found in the future works section. The initial design implemented the use of an additional microcontroller to serve as a watchdog to prevent system runaway should the Raspberry Pi fail, this is also addressed in the future works section of this report. For the final design choice, the team chose to use a single microcomputer for sending commands to the necessary components when needed. Python 3.9 was used as the programming language for its variety in libraries, especially for the spectrophotometer as open source software for the hardware was available.To establish connection with the communication board, the team used the python library PySerial.

From the documentation on the communication board, it was noted that data can be encoded and sent from a windows machine via USB. This was tested using a program called coolTerm, a similar terminal program exists on the Linux based MCU's called Miniterm, as part of the PySerial library. It was possible to write commands in the program and send it to the communication board. The communication board then sends commands to a motor to execute. In order to establish a

connection with the communication board using the MCU, pySerial was chosen according to the research made on serial connections.

Using python, a pySerial object was used to write data out of a designated USB port. First the address to the USB port connected to the communication board was found and using pySerial an object was created that could be used to write and read data from the port. The object was then configured to encode the data sent out in hex, as the communication board required. The final step was writing the commands for the motors, and encoding it in bytes UTF-8 so that the motor could recognize the commands after the serial to RS485 conversion.

In order for the system to collect data, a spectrophotometer made by Ocean Optics, the Flame-S-UV-VIS is used with the system to measure the intensity of light through the reagent-sample mix. The spectrophotometer records the intensities of light in "arbitrary units … [depending] on the ADC bit resolution of the hardware used in the specific spectrometer"[17]. The units of intensity of this spectrophotometer is accounted for and the setup is done by the MLML, providing the proper parameters for initializing the spectrophotometer. Once initialized, the spectrophotometer is controlled with a python library called Seabreeze.

Seabreeze is an open source library that can be used to collect the intensities recorded by the spectrophotometer over its operating range of wavelengths. The python program includes initializing the device connection through USB and obtaining the intensities and wavelengths recorded by the spectrometer. This information is used for calculations to obtain the absorbance of the reagent-sample mix. By taking the absorbance of a deionized water, a sample of known PO4 concentration, and the mix,the concentration of PO4 can be calculated.

Software Design

This project's aim is to design and construct a nutrient analyzer that will sample ocean water from a continuous source every hour for thirty days. The process of measuring for nutrients requires deionized water and reagents ascorbic acid, molybdate, mixed and measured at specific ratios and times. The measurement is made using a spectrometer so the device must be flushed between measurements to ensure accuracy of results.

Absorbance is scalar units as it is a value with respect to another measurement. This is essential to taking measurements using a process such as this that is time and temperature sensitive. With this set up, the spectrophotometer is calibrated each using a sample of a known concentrate and deionized water.

The priority of the project is to obtain precision motor control through a microcomputer. The objective is to synchronize the action of the motors and control flow rates to precisely mix the reagents. Following this, the next objective will be to mix the chemicals with the sample water and pump the mixture into a flow tube where the spectrophotometer will measure the light going through the mixture. This gives an absorbance reading used to calculate the concentration of the targeted nutrient in the sample. Following the successful data acquisition, the system is flushed with water in order to prevent inaccurate measurements and contamination.

Figure 10

Benchtop setup while in operation

The primary system consists of two holding coils, each connected to a motor to act as a pump to both aspirate and dispense. The pumps connect directly to a deionized water reservoir as well as the holding coil. The other end of each holding coil is connected to a rotating piping system that

allows the pump to aspirate from a variety of fluids including molybdate, ascorbic acid, and the sample. A short description of the target program procedure is as followed:

1) The system will perform a flush with deionized water to ensure that remaining chemicals are washed out.

2) A dark scan (no light, flow tube filled with deionized water) is performed.

3) Light is turned on and a reference scan (light on, deionized water) is performed.

4) The rotating assembly will connect pump/holding coil 1 to reagent 1 and aspirate x amount of the reagent.

5) The rotating assembly will connect holding coil 2 to reagent 2 and aspirate x amount of reagent. Simultaneously, the rotating assembly should have holding coil 1 and 2 connected and pump the reagent 1 in coil 1 to coil 2 to mix the chemicals.

6) The rotating assembly will connect holding coil 1 to the sample and aspirate x amount of the sample. Simultaneously, the assembly should have holding coil 1 and 2 connected and pump the mixture from coil 2 into coil 1.

7) The rotating assembly will connect holding coil 1 to the flow tube and pump the final mixture into the tube.

8) A sample scan (final mixture) is performed. The absorbance is then calculated with the dark and reference scan.

7) The system will perform a flush

Key elements of the programming include ensuring the proper amounts of each fluid are aspirated and that the reagents are continuously with deionized water. Additionally, the pumps must be synchronized to prevent pumping unintended fluids into the reservoirs creating contamination. The pumps must also be precise so that the proper chemical reaction will occur.

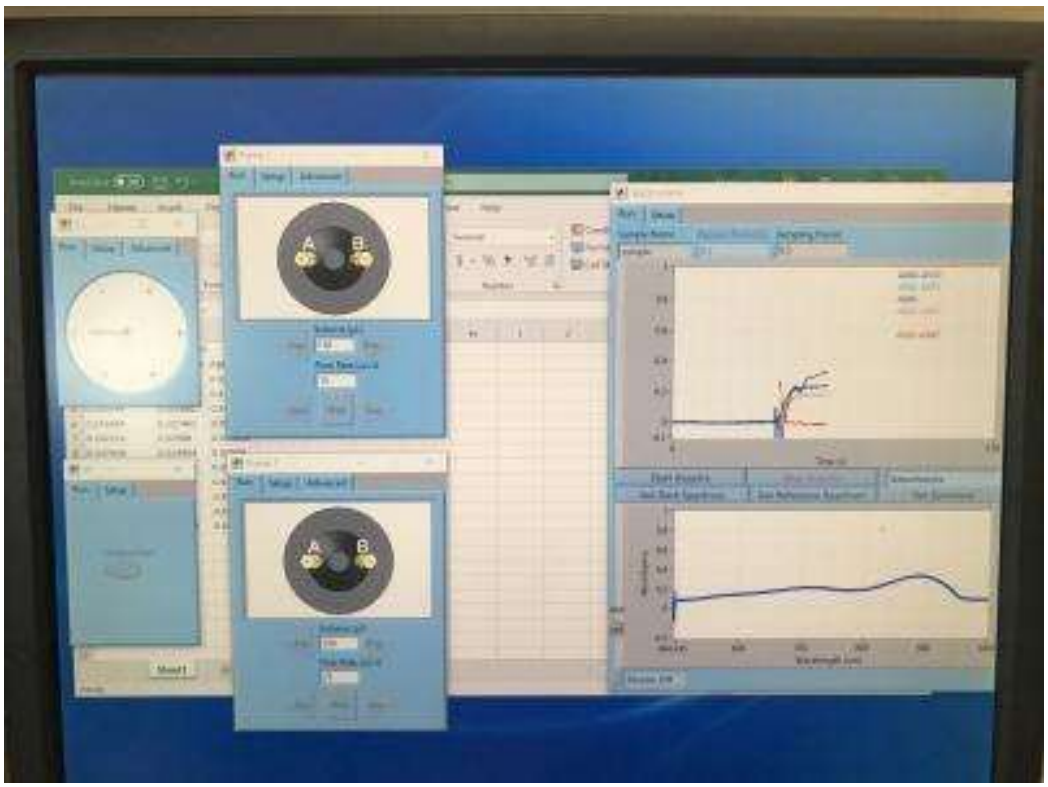
Python software was engineered to perform controlled aspirating and dispensing of the pumps. The rotary valve motor was programmed to move to the necessary positions to control the flow

direction between the pump and reagents/reactants. The functions in the python will include calls to the motors of pump 1, pump 2, and the rotary valve to actuate them.

The python program will receive inputs from the spectrophotometer and store the values in an array data structure. The team worked with a SJSU CMPE division to have the analyzer communicate wirelessly with the MLML computers.

Figure 11

Graphic user interface for controller pump speed and output from absorbance value time series taken in real time.



Chapter 4

Software Implementation Results

The software that was created using python 3.9 was a successful endeavor. Running through the script from top down, the first area consists of end-user variables (Appendix C1: Controlled Variables). Here all user controlled inputs are declared. This was done to allow the user to have clear and concise variable control over the script. For the port position on the valve one can easily change the position which will reflect changes directly to the port valve movement functions(Appendix C2: Chem On Valve Servo Control). The user also has control over pump speed and direction based on the user's needs. This was an important implementation, due to possible changes in chemical assays in the future. By simply manipulating the speed and amount needed to be dispensed or aspirated, the function variables for any pump movement functions are automatically updated (Appendix C3: MilliGat Pump Control). Additionally, with respect to the pump control, the timing sequence desired for the pump movement to take place was also taken into consideration. Time control variables were implemented with simple calculation using the absolute values of pump one movements (dispense is + and aspirate is -). By taking the amount desired to move divided by speed at which to move, the time required for the movement is the result. Therefore anytime the user decides to change the amount and speed of any dispensing functions the time required is automatically updated, again allowing for end-user ease of control(Appendix C1: Controlled Variables). One of the other areas that was successfully achieved was the priming sequence. User requested setting up a priming run for anytime reagents are changed in the field to ensure proper instrumentation set up and initialization Priming variables were established with the same ease of control in mind so that the user only need to update the pump priming variables (Appendix C1: Controlled Variables) which in turn update the pump priming directive functions (Appendix C5: Pump Priming). The spectrometer control was required to do four different functions, a darkscan, reference scan, sample scan, and an absorbance calculation (Appendix C4: Seabreeze Spectrometer Control). These were existing libraries for the Seabreeze Optics Flame spectrometer and used to create the required functions for collecting the intensities and wavelengths, then using that to calculate the absorbance of the control sample and unknown sample to determine the concentration of phosphate with respect to each (these results will be discussed later in the test results section). Additionally, auxiliary motor and LED lamp

control was successfully implemented (Appendix C7: Aux Motor And LED Lamp Control) using the Raspberry Pi4's Gpio pins. This allowed for the aux motor to pull unknown samples from the pump house at Moss Landing to fill a sample container, which is then used by servo and pump control to implement analysis autonomously on the unknown sample. LED lamp control was established by turning the LED lamp on and off as required by different spectrometer scanning parameters. Lastly, the sequence that was provided by MLML (Appendix C9: Insitu PO4 Sequence) was created into three separate run functions (Appendix C6: Moss Landing Sequence Control) which in turn was then used to create the final while loop control to make the system run infinitely (Appendix C8: Autonomous Control Cycle) and continuously record spectrometer values and designated times to collect data from the system. With over thirty functions created to run the system, implementation of end-users requirements was achieved and successfully tested. Furthermore, the users requirements of a software that was easy to read, understand, and change parameters with minimal interaction was also achieved.

Test Results

When running the system it was done in a sequential format under the guidance of the Moss Landing Marine Lab's Chemical Oceanography department. Firstly, the blank_sample_run code() was run from Appendix C6. The data collected from fig.12 shows the intensity vs. wavelength data of the dark scan, which yielded the expected value pattern of a flat line. Since the substance in the flow tube is deionized water, it is expected to see a flat line as there is no direct light read. The values given by the chart are used to zero the error when calculating the absorbance. The next set of data obtained is the reference scan (fig.13). This shows the values of the intensities when the light is shone through the water and is used as a reference for the samples scan for absorbance calculation. Finally a sample scan is performed, which shows the intensities across wavelengths of the light passing through the sample(fig.14).The absorbance is then calculated, shown in fig.15. When performing absorbable calculations for a blank, PO4 standard sample (fig.16), and the sea water sample (fig.17) using a dark, reference, and sample scan for each, it is shown that the seawater values for absorbance falls between the blank and the PO4 standard. The MLML states that the results are to be expected and fall within reason. Calculating the PO4 concentration in the sample using Eq.(3), a value of .6 umol/kg was obtained. As there is no theoretical value, as the concentration in the sample will vary from sample to sample, the quality of the measurement is

determined within reason of the knowledge of the chemistry department of MLML. The MLML team finds our measurements to be reasonable and verify that the instrument works to appropriate standards.

Figure 12

Dark scan, measurement of intensity with pure deionized water and no light source.

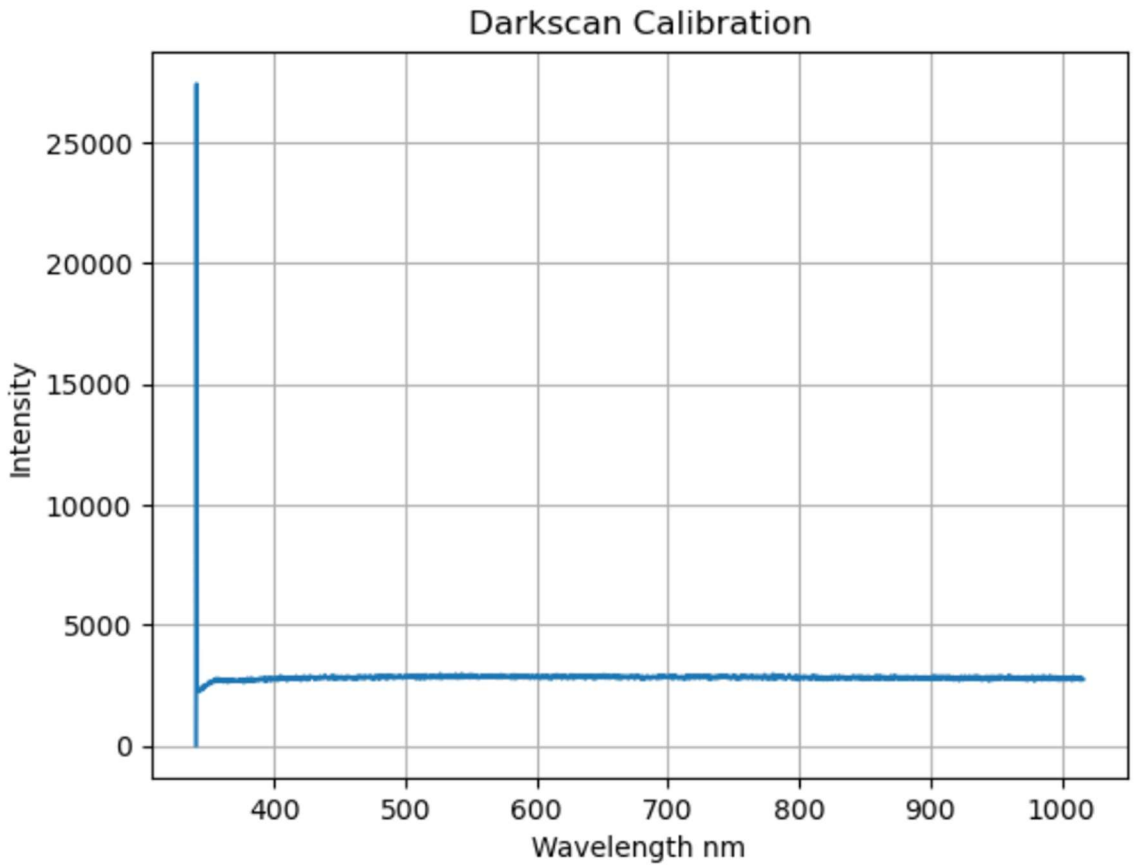


Figure 13

Reference scan, intensity of light shone through pure deionized water.

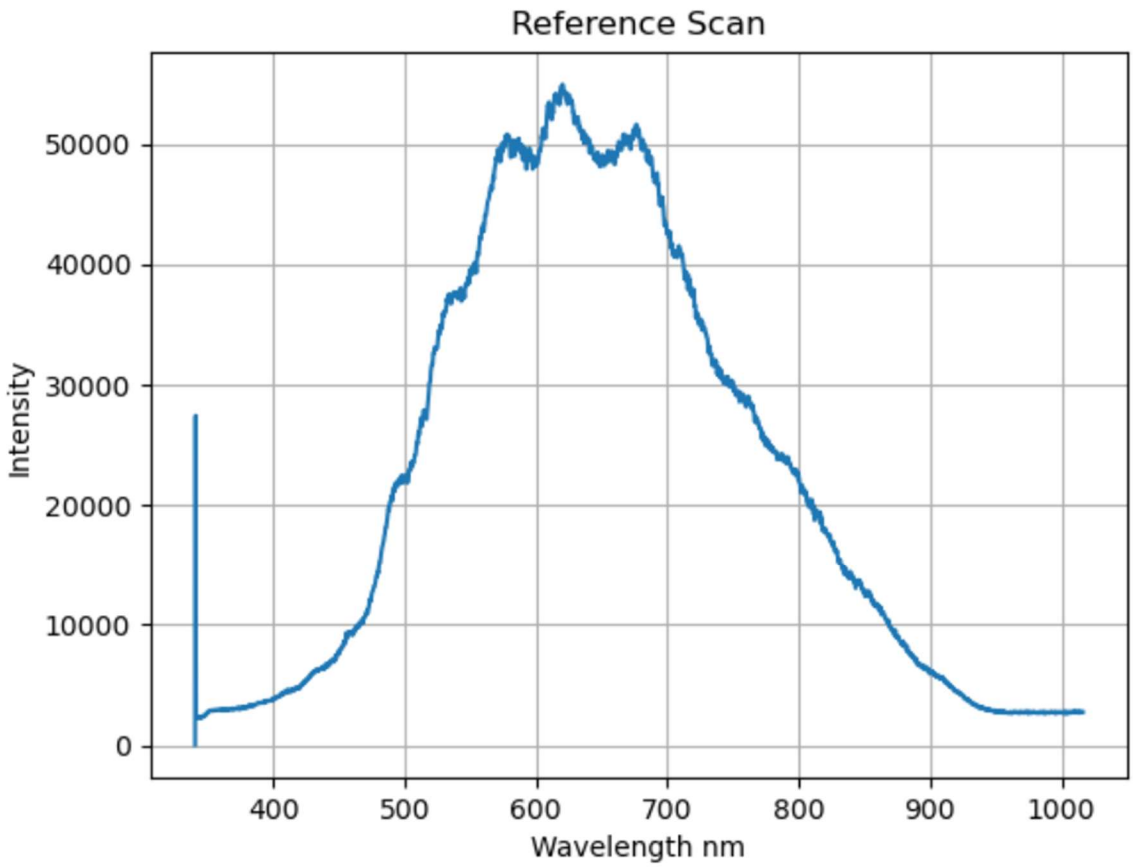


Figure 14

Sample scan, graph of intensity of light shone through the reagent and sample (blank depicted) mix

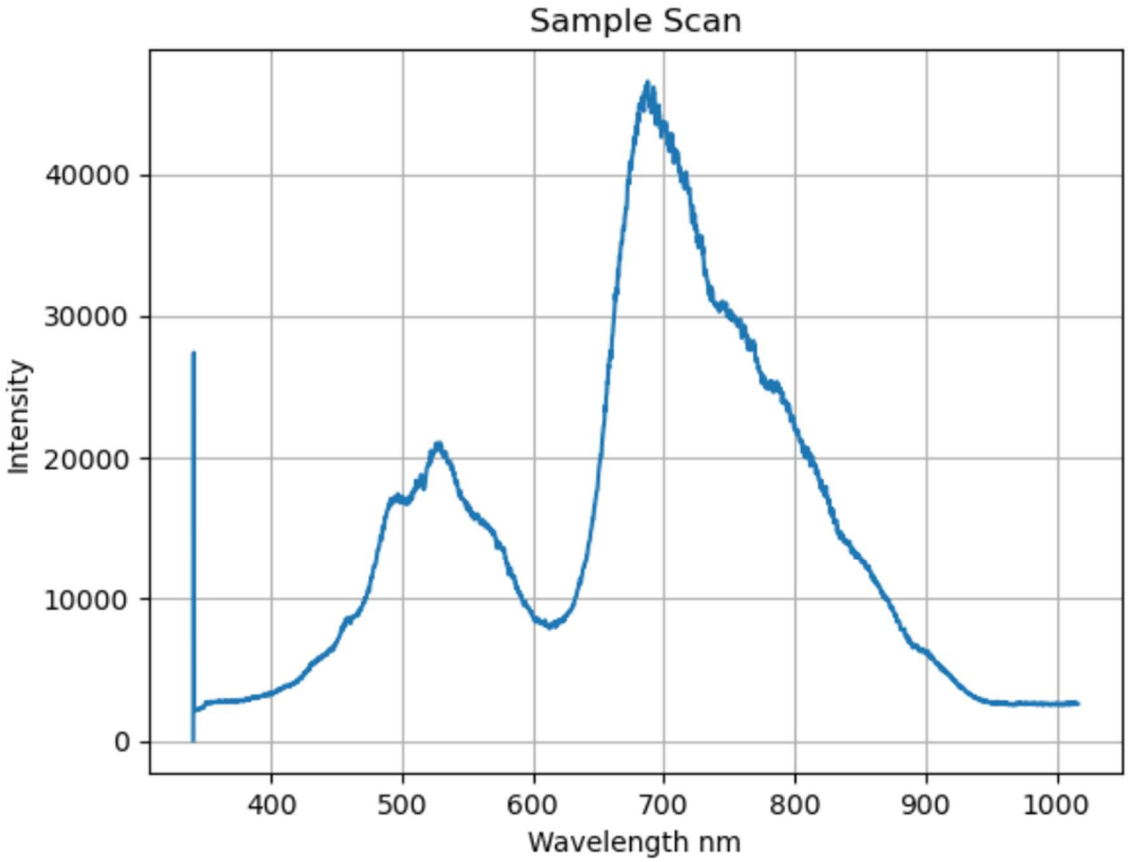


Figure 15

Blank (deionized water+reagents)absorbance value, calculated from a reference, dark ,and blank sample intensities.

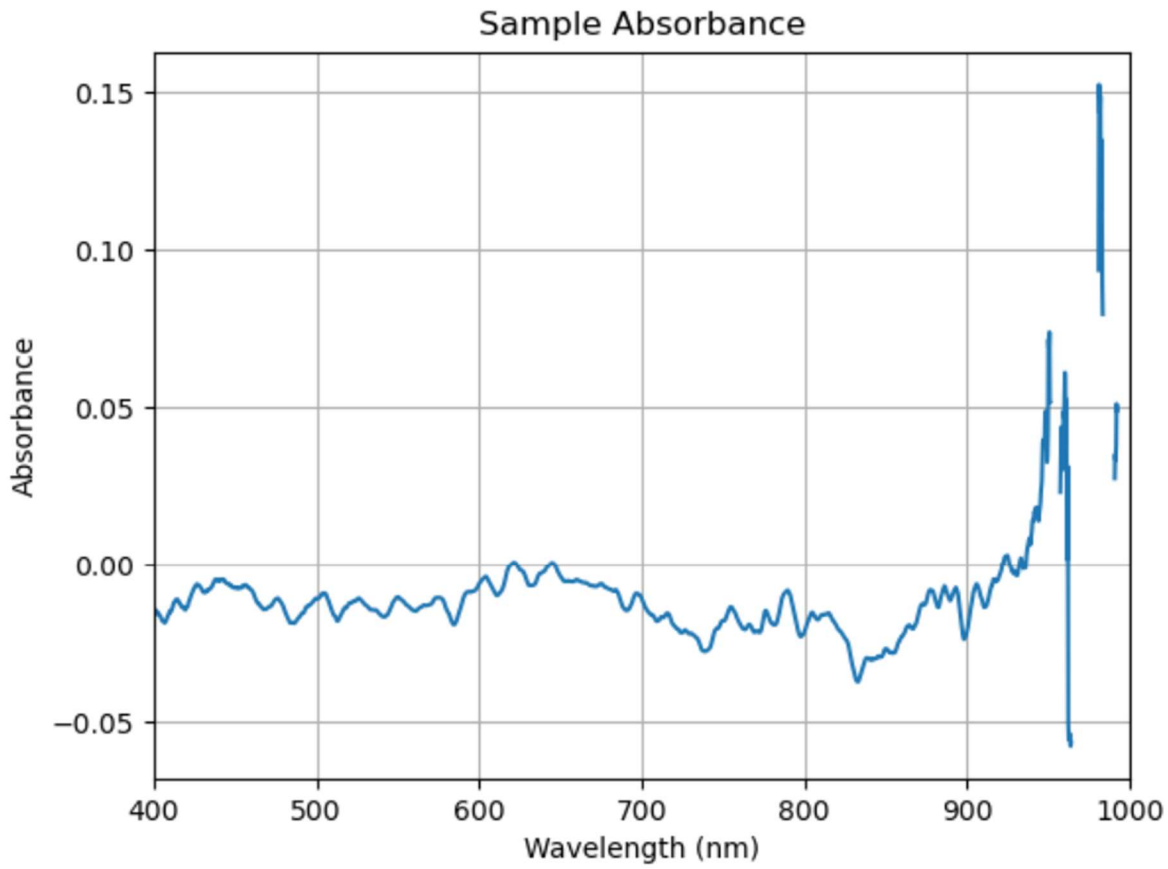


Figure 16

PO4 Standard (diluted PO4 solution) absorbance value, calculated from a reference, dark ,and PO4 sample intensities.

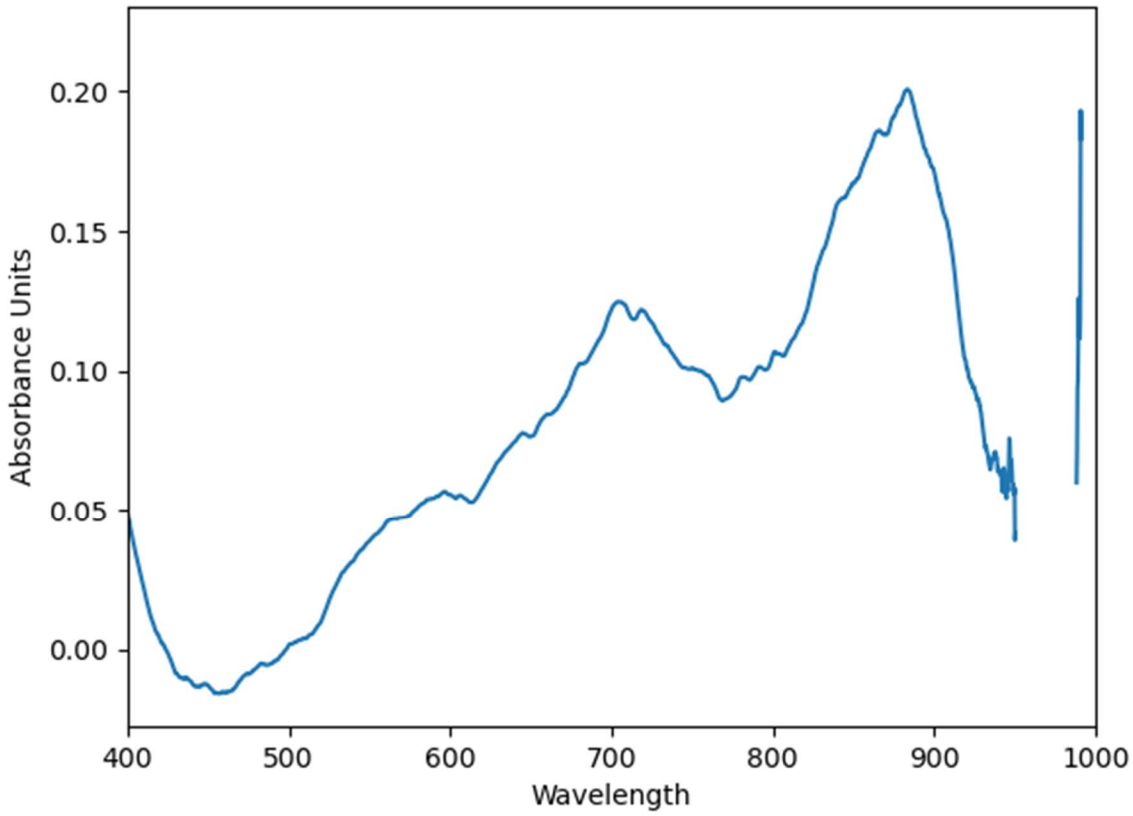
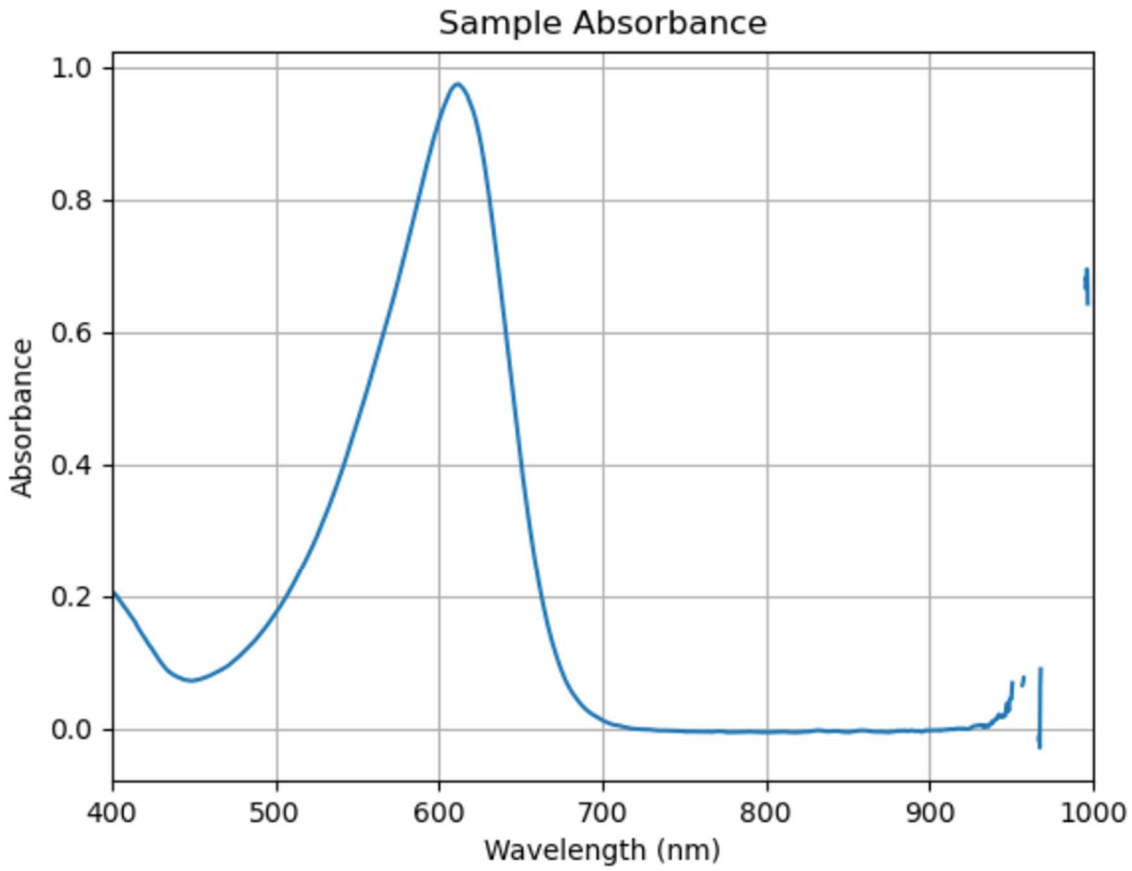


Figure 17

Sample (Sea water) absorbance value, calculated from a reference, dark ,and Sample intensities.



Chapter 5

Conclusions

By combining the components and implementing relevant software, a successful prototype was completed with a build price less than half of current benchtop systems used in the Moss Landing Marine Lab. By referencing documentation on the hardware and studying communication protocols, successful control with a microcomputer system was implemented. Following sequence guidelines from the marine lab, an autonomous program was designed and completed with correct hardware control.

The MLML finds the data collection of the system to be acceptable and within range of expected values. Hardware functions correctly and data collection and processing is consistent without crashing.

The system is currently being integrated with CMPE software for wireless access and gui. Wireless communication and data transfer is still yet to be fully implemented.

The system is able to autonomously run data collections every hour and store the necessary data for analysis by the MLML team. The system has completed over 100 consecutive runs of data collection (pure water runs, as the ME team does not have access to the required chemicals) and is proven to be robust as the actual implementation will see one cycle once an hour. MLML is satisfied with the prototype presented.

Recommendations for Future Works

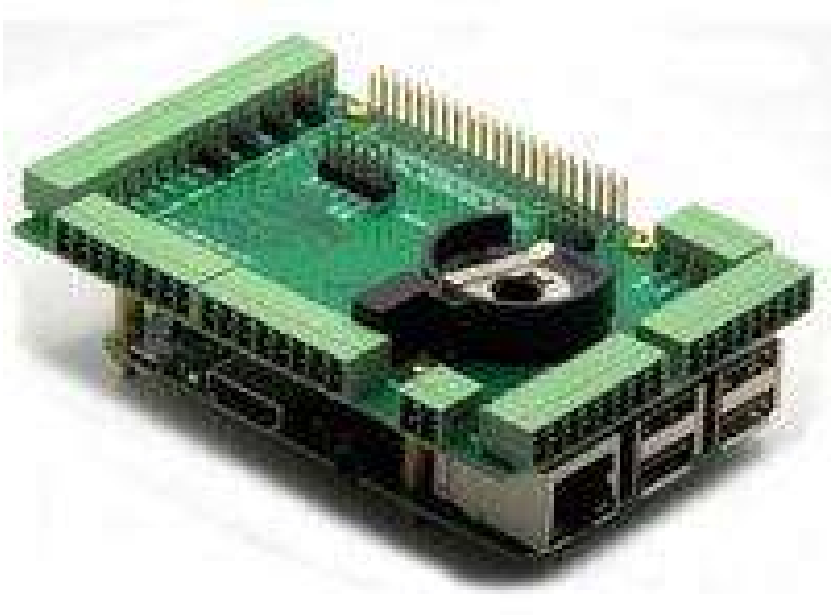
This device has only begun to be calibrated and the interface of it is still in the process of being developed. In many ways this project is still being created. The following suggestions are for the use and continued improvement of pFIONA. In future versions it would be beneficial to externalize the power supply from the instrument and include an estimated 1 Watt/hour battery or higher in order to allow the device to flush itself in case of a power failure.

Additionally it would be useful to include a watchdog microcontroller to perform a heartbeat function of the device and potentially serve as the device's interface. There is potential for this device to be wireless and capable of monitoring several pFIA devices at a time. This MCU would communicate with the Raspberry Pi either through bluetooth or I2C communication. A feather board with bluetooth capabilities is suggested to be paired with a small E-Ink screen to serve as a low power graphic display capable of being used as a basic interface for the user as well as a status and finding graph representation.

Figure (18) below shows the suggested Raspberry Hat that could be used to eliminate the proprietary communication PCB from the Instrument, lowering the overall cost and reducing the size of the enclosure necessary. This hat from Sequent Microsystems provides serial to RS485 conversion and transmission. It also provides controllable 5 volt and 24 volt power distribution. This hat is designed to be stack-able. Meaning one Raspberry Pi could be paired with up to eight of these hats and control simultaneously up to 8 pFIAONAs.

Figure 18

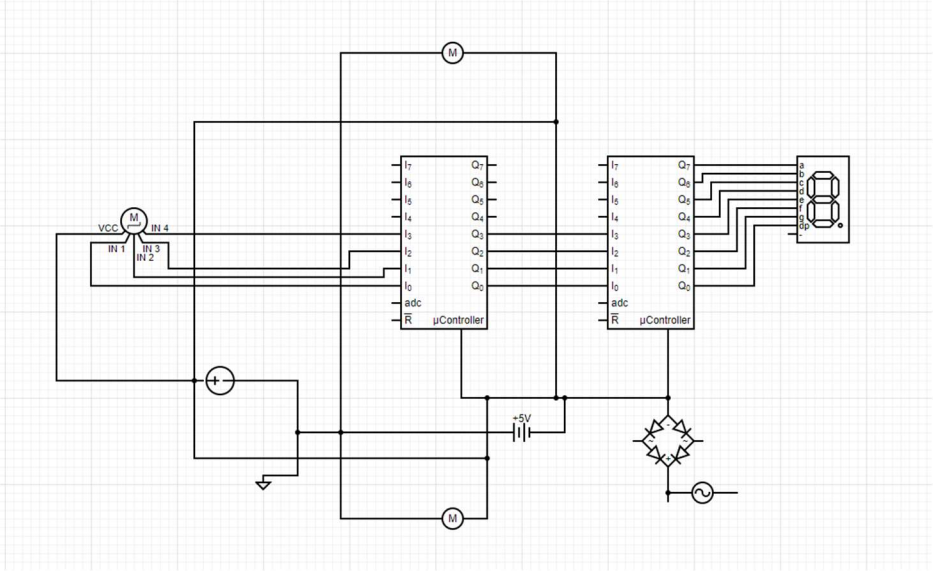
Sequent Microsystems Industrial Interface Hat for Raspberry Pi.



The suggested use of a watchdog / heartbeat additional MCU is shown below in figure (19). This secondary MCU can be as simple as a $4 pico and still perform the heartbeat function. However, do to the lack of backup power supply in the instrument an additional heartbeat or watchdog MCU could not function without the main power supply and thus does not provide additional protection presently

Figure 19

Circuit diagram of mechanical control system.



The following feedback systems are suggested to ensure the process does not get contaminated are planned to be as follows: First and foremost a sensor to stop the cycle before any of the chemical compounds runs out. Secondarily, a small backup power supply of an estimated 1 watt hour in order for the instrument to be able to flush the holding coils and servo valve in order to preserve the sample and not contaminate the device in the event of a power disruption or outage. If possible a tertiary feedback for signaling when a sample is out of the expected range to notify the user of a potential problem with the system by flagging the data and the timestamp of the event. Additionally, it would be useful to include the current text prompts in the program's sequence as a baseline state machine for knowing what the instrument is doing and where it has failed should it run into a problem.

Global Impacts

The international implementation of devices such as this will present more and more people with the ability to probe into their environment. There is a wide range of potential experiments to be run using a pFIA device such as pFIONA. For now the project has helped to democratize water analysis by lowering financial costs and technical barriers to entry from science. In place such as the project's intended use, a deeper understanding of the human impact on the environment from industries such as agriculture on environments. A finer detail and resolution for PO4 levels over time will allow for understanding of what causes them and how they can be mitigated to lessen their impact on ecosystems.

References

[1] *MBNMS Resource Issues: Ecosystem Conservation and Biodiversity Protection*. (n.d.). Montereybay.noaa.gov Retrieved September 25, 2021, from https://montereybay.noaa.gov/resourcepro/resmanissues/biodiversity.html

[2] USGS. (n.d.). *Phosphorus and Water*. Usgs.gov. https://www.usgs.gov/special-topic/water-science-school/science/phosphorus-and-water?qt-science_center_objects=0#qt-science_center_objects

[3] Hallegraeff, G. M. (1993). "A Review of Harmful Algal Blooms and Their Apparent Global Increase." Phycologia, https://doi.org/10.2216/i0031-8884-32-2-79.1.

[4] *Monterey County Farm Bureau - Facts, Figures & FAQs*. (n.d.). Montereycfb.com. http://montereycfb.com/index.php?page=facts-figures-faqs

[5] *Nutrient analysers - Coastal Wiki*. (n.d.). Www.coastalwiki.org. Retrieved September 25, 2021,http://www.coastalwiki.org/wiki/Nutrient_analysers#:~:text=A%20variety%20of%20wet%20chemical%20nutrient%20analyzers%20exist

[6] *Phosphorus and Your CKD Diet*. (2018, November 2). National Kidney Foundation. https://www.kidney.org/atoz/content/phosphorus

[7] *Nutrient analyzers and sensors*. (n.d.). Www.endress.com. Retrieved December 01, 2021,

https://www.endress.com/en/field-instruments-overview/liquid-analysis-product-overview/nutrient-analyzers-sensors#:~:text=Nutrient%20analyzers%20and%20sensors%20are%20used%20in%20the

[8] *Systems: miniSIA-2*. (n.d.). Www.globalfia.com. Retrieved December 01, 2021, from https://www.globalfia.com/store/view/productdetails/virtuemart_product_id/200/virtuemart_category_id/7

[9] Ibnul, N. K., & Tripp, C. P. (2021). A solventless method for detecting trace level phosphate and arsenate in water using a transparent membrane and visible spectroscopy. *Talanta (Oxford)*, *225*, 122023–122023. https://doi.org/10.1016/j.talanta.2020.122023

[10] Legiret, F.-E., Sieben, V. J., Woodward, E. M. S., Abi Kaed Bey, S. K., Mowlem, M. C., Connelly, D. P., & Achterberg, E. P. (2013). A high performance microfluidic analyser for phosphate measurements in marine waters using the vanadomolybdate method. *Talanta (Oxford)*, *116*, 382–387. https://doi.org/10.1016/j.talanta.2013.05.004

[11] Jarda Ruscika. (2021). *Flow Injection Tutorial*. Flowinjectiontutorial.com. https://www.flowinjectiontutorial.com/Theory%200.1.1.%20Introduction%20-%20From%20Batch%20to%20Flow%20Based%20Analysis.html

[12] *Gravimetric Analysis Principle with Types, Advantages and Examples*. (n.d.). BYJUS.

Retrieved December 01, 2021, from https://byjus.com/chemistry/gravimetric-analysis/#:~:text=The%20principle%20of%20Gravimetric%20Analysis%3A%20The%20principle%20behind

[13] *Titrimetry - an overview | ScienceDirect Topics*. (2019). Sciencedirect.com. https://www.sciencedirect.com/topics/chemistry/titrimetr

[14] 2.5: Reaction Rate. (2013, October 2). Chemistry LibreTexts. https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Kinetics/02%3A_Reaction_Rates/2.05%3A_Reaction_Rate#:~:text=The%20Reaction%20Rate%20for%20a%20given%20chemical%20reaction

[15] Principles of Flow Injection Analysis. (2020, January) Researchgate. https://www.researchgate.net/publication/338435099_Principles_of_Flow_Injection_Analysis

[16] Spectrophotometer. N.D. labcompare.com https://www.labcompare.com/Spectroscopy/106-Spectrophotometer/

[17] Poehlmann, A. *API Reference Python Seabreeze.* https://python-seabreeze.readthedocs.io/en/latest/api.html

[18] Liechti, C. *pySerial API.* https://pyserial.readthedocs.io/en/latest/pyserial_api.html

Appendices

Appendix A: Gantt Chart

Figure 20

Original Gantt Chart

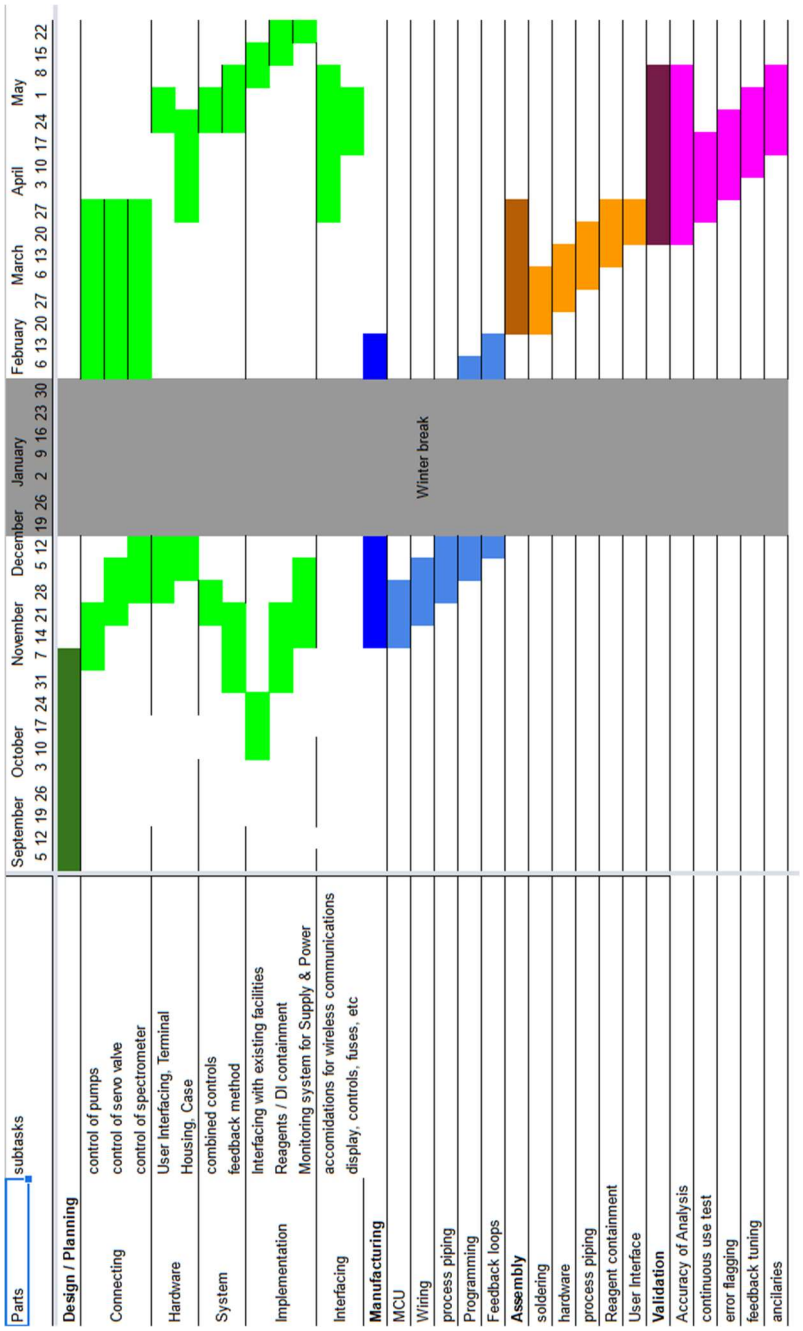
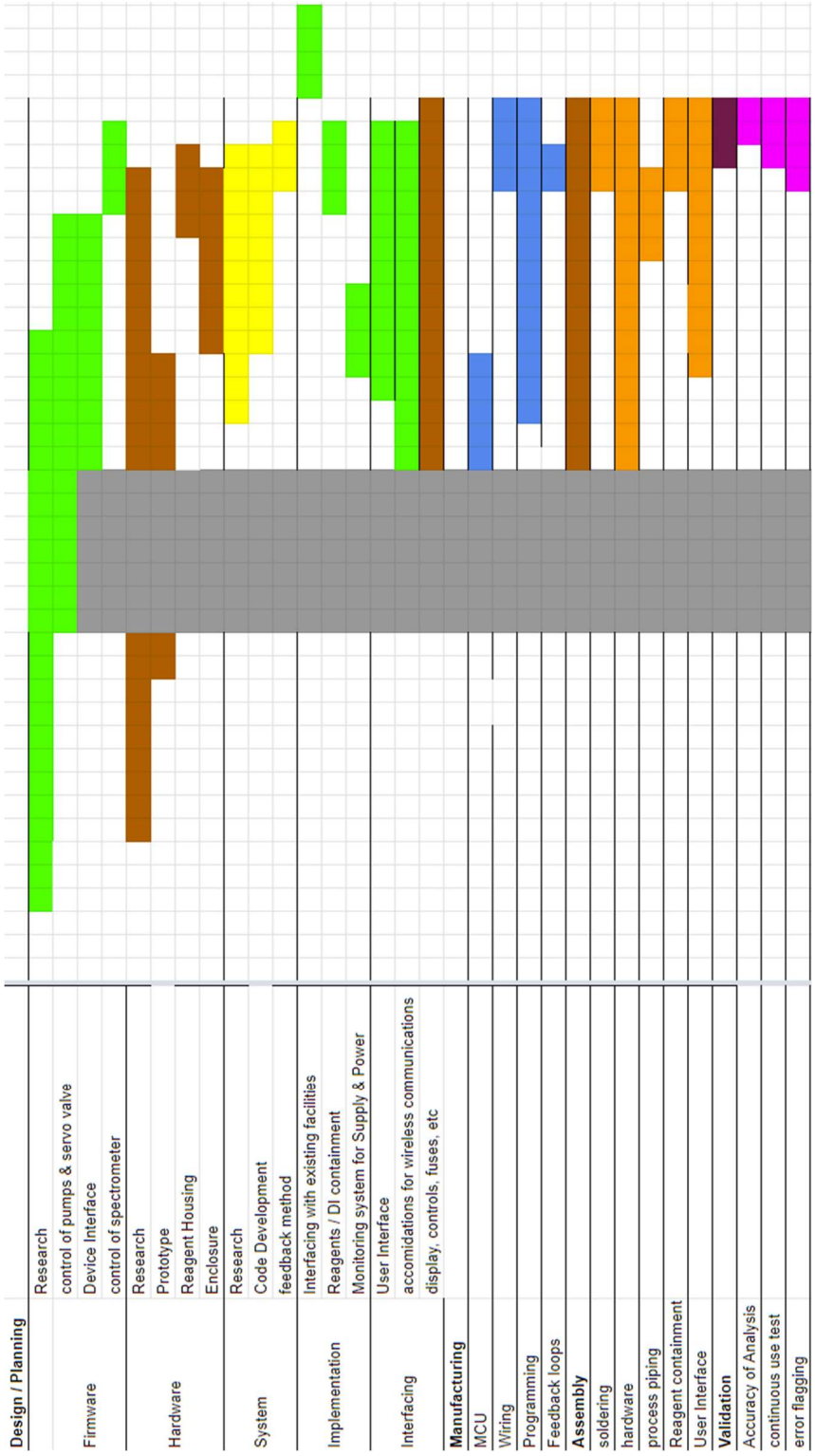


Figure 21

Actual Gantt Chart for the project



Appendix B: Various Elements of Benchtop version

Figure 22

A picture showing the holding tube for spectrometer analysis

Figure 23

Top view of the instrument showing the two pumps and holding coils wrapped around aluminum cylinders.

Figure 24

Front view of apparatus depicting the servo driven multiport valve.

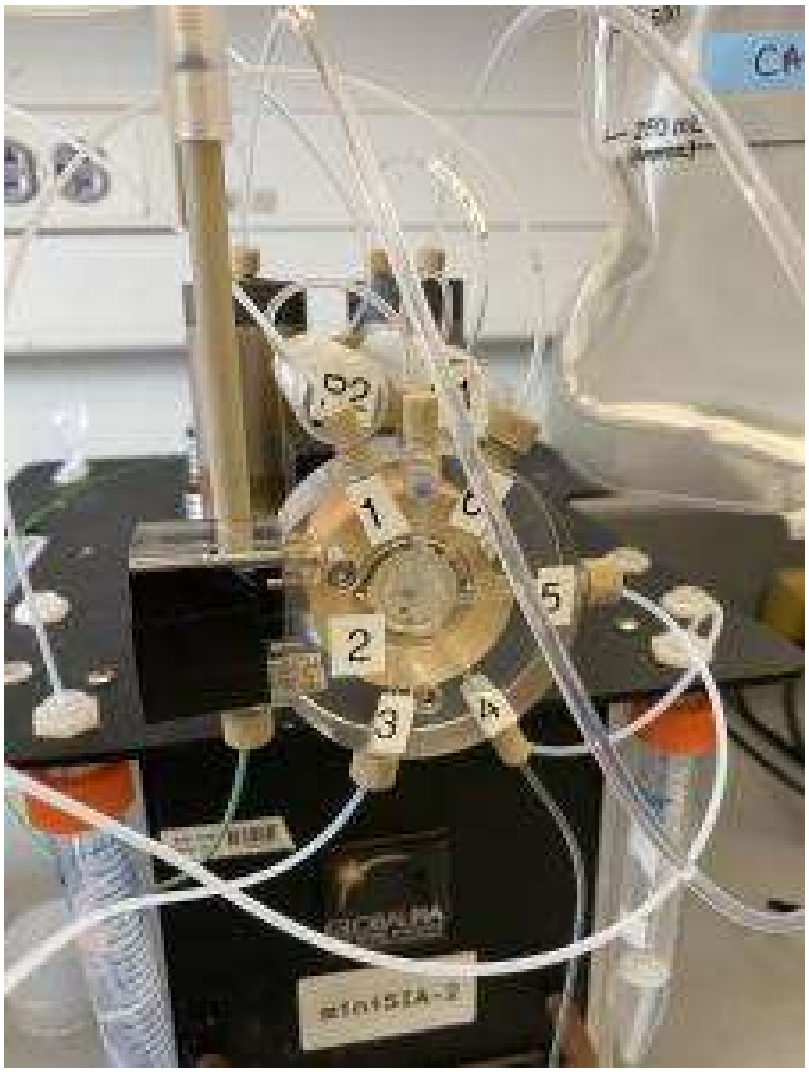


Figure 25

Side view showing mounting of pumps and rotary valve.

Figure 26

Original layout of the miniSIA-2 components from GlobalFIA

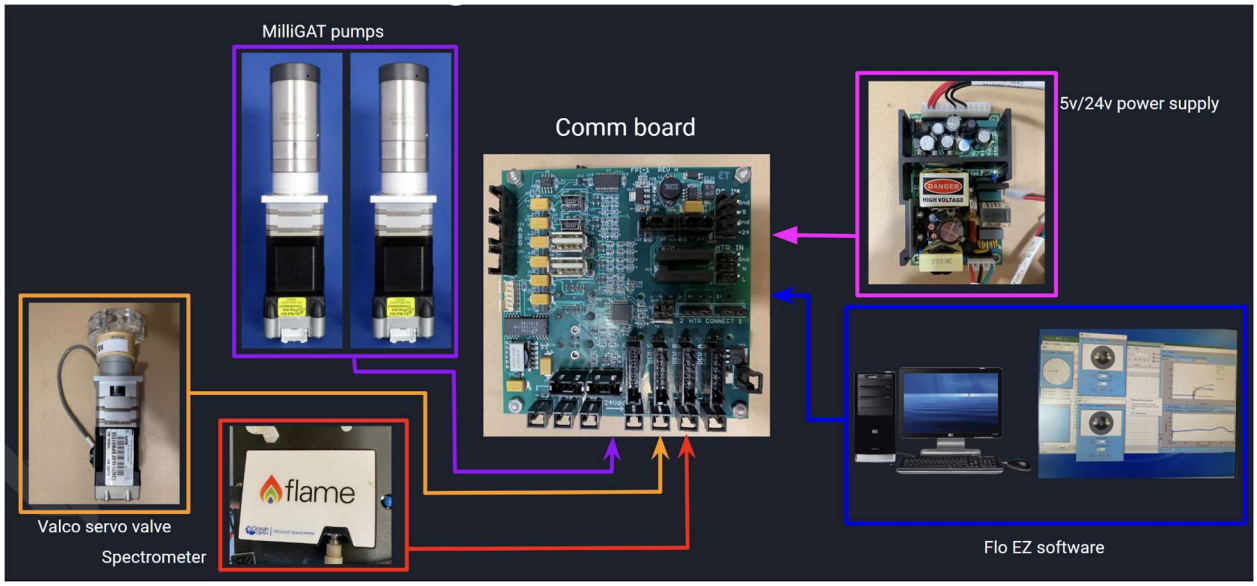
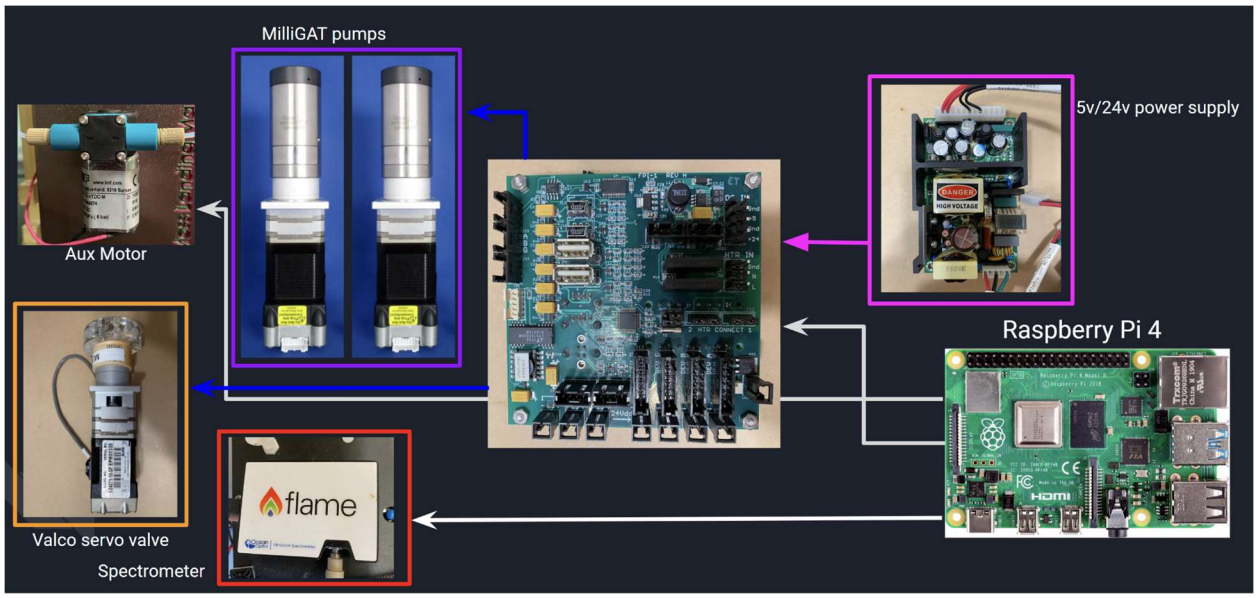


Figure 27

Final design concept utilized in pFIONA construction



Appendix C: Code

Appendix C1: Controlled Variables

```
                # SPECTROMETER INITIALIZATION VARIABLES

from seabreeze.spectrometers import Spectrometer
spec = Spectrometer.from_first_available()
spec.integration_time_micros(50000) ##integration time in microseconds

#define monitoring and reference wavelengths to be used
monitoringlambda = 880
reflambda = 600

known_concentration = 2




        #CHEM VALVE PORT VARIABLES

#Port Positions
port_p1 = 1340                  # Waste Port position (Currently not in use)
port_p2 = 1090                  # Flow Cell position
port_p3 = 840                   # Molybdate position
port_p4 = 590                   # PO4 Sample position
port_p5 = 350                   # Ascorbic Acid position
port_p6 = 130                   # PO4 Standard position

        # PUMP SPEED VARIABLES
#System Flush
flush_p1_speed = 150            # pump 1 speed in uL/sec
flush_p1_amount = 1000          # pump 1 amount in uL (+ for dispense, - for aspirate)
flush_p2_speed = 150            # pump 2 speed in uL/sec
flush_p2_amount = 1000          # pump 2 amount in uL (+ for dispense, - for aspirate)

#Blank Sample
blank_p1_speed = 60             # pump 1 speed in uL/sec
blank_p1_amount = 600           # pump 1 amount in uL (+ for dispense, - for aspirate)

#Molybdate Reagent
molybdate_p1_speed = 32         # pump 1 speed in uL/sec
molybdate_p1_amount = 320       # pump 1 amount in uL (+ for dispense, - for aspirate)
molybdate_p2_speed = 40         # pump 2 speed in uL/sec
molybdate_p2_amount = -400      # pump 2 amount in uL (+ for dispense, - for aspirate)

#Ascorbic Acid Reagent
ascorbic_p1_speed = 40          # pump 1 speed in uL/sec
ascorbic_p1_amount = -400       # pump 1 amount in uL (+ for dispense, - for aspirate)
ascorbic_p2_speed = 32          # pump 2 speed in uL/sec
ascorbic_p2_amount = 320        # pump 2 amount in uL (+ for dispense, - for aspirate)

#Flow Cell
flow_cell_p1_speed = 25         # pump 1 speed in uL/sec
flow_cell_p1_amount = 400       # pump 1 amount in uL (+ for dispense, - for aspirate)

#PO4 Standard
po4_standard_p1_speed = 60      # pump 1 speed in uL/sec
po4_standard_p1_amount = -600   # pump 1 amount in uL (+ for dispense, - for aspirate)

#PO4 Sample
po4_sample_p1_speed = 60        # pump 1 speed in uL/sec
po4_sample_p1_amount = -600     # pump 1 amount in uL (+ for dispense, - for aspirate)

        # PUMP PRIMING VARAIBLES

totalprimes = 2                 # indicates number of times prime cycle should repeat (set by end-user)

port1_pp1speed = 100            # port 1, pump 1, priming speed in uL/sec
port1_pp1amount = 1000          # port 1, pump 1, priming amount uL (+ for dispense, - for aspirate)

port2_pp1speed = 150            # port 2, pump 1, priming speed in uL/sec
port2_pp1amount = 2000          # port 2, pump 1, priming amount uL (+ for dispense, - for aspirate)
port2_pp2speed = 150            # port 2, pump 2, priming speed in uL/sec
port2_pp2amount = 2000          # port 2, pump 1, priming amount uL (+ for dispense, - for aspirate)

port3_pp1speed = 100            # port 3, pump 1, priming speed in uL/sec
port3_pp1amount = -200          # port 3, pump 1, priming amount uL (+ for dispense, - for aspirate)

port4_pp1speed = 100            # port 4, pump 1, priming speed in uL/sec
port4_pp1amount = -200          # port 4, pump 1, priming amount uL (+ for dispense, - for aspirate)

port5_pp1speed = 100            # port 5, pump 1, priming speed in uL/sec
port5_pp1amount = -200          # port 5, pump 1, priming amount uL (+ for dispense, - for aspirate)

port6_pp1speed = 100            # port 6, pump 1, priming speed in uL/sec
port6_pp1amount = -200          # port 6, pump 1, priming amount uL (+ for dispense, - for aspirate)
```

```
                # TIME VARIABLES

t0 = 0.001                          # time for running pumps simutaneously
t1 = 3                              # sleep time for valve movement and pump pause ( may be adjusted to suit end-user needs)
t3 = 30                             # flow cell wait time for absorbance measurement (set based on end-user needs)
auxtime = 5                         # aux pump time for drawing sample seawater to refill sample (set based on end-user needs)
darkscantime = 5
refscantime = 5
flushtime = abs(flush_p1_amount/flush_p1_speed)+t1                          # This area calculates the time needed for pumps to run.
blanktime = abs(blank_p1_amount/blank_p1_speed)+t1                          # It takes the absolute value of the pump 1 ratio of amount
molybdatetime = abs(molybdate_p1_amount/molybdate_p1_speed)+t1              # of volume to move and the speed at which the volume is moved.
po4sampletime = abs(po4_sample_p1_amount/po4_sample_p1_speed)+t1            # This is done to ensure accurate numbers due to dispensing and
po4standardtime = abs(po4_standard_p1_amount/po4_standard_p1_speed)+t1      # aspirating values being +-. This value is then added to sleep
ascorbicacidtime = abs(ascorbic_p1_amount/ascorbic_p1_speed)+t1             # time for valve movement and pump pause to ensure adequate time
flowcelltime = abs(flow_cell_p1_amount/flow_cell_p1_speed)+t1               # for sequence to finish.
p1primetime = abs(port1_pp1amount/port1_pp1speed)+t1
p2primetime = abs(port2_pp1amount/port2_pp1speed)+t1
p3primetime = abs(port3_pp1amount/port3_pp1speed)+t1
p4primetime = abs(port4_pp1amount/port4_pp1speed)+t1
p5primetime = abs(port5_pp1amount/port5_pp1speed)+t1
p6primetime = abs(port6_pp1amount/port6_pp1speed)+t1

                #AUX MOTOR AND LIGHTSOURCE PIN CALLOUT

motorPin=17                         # Define Pins for aux motor and lightsource
ledPin=27

GPIO.setmode(GPIO.BCM)
GPIO.setup(motorPin,GPIO.OUT)       # Set aux motor pin to output
GPIO.setup(ledPin,GPIO.OUT)         # Set lightsouce to ouput
GPIO.output(motorPin,GPIO.LOW)      # Aux motor output initial state of 0
GPIO.output(ledPin,GPIO.LOW)        # Lightsource output initial state of 0
```

Appendix C2: Chem On Valve Servo Control

```
                #CHEM on VALVE PORT POSITION FUNCTIONS

def port_1():
        print("Moving to port 1")

        port_1_pos ='AMA '+ str(port_p1)+'\r\n'         #Uses port_1 variable. AMA is absolute position from 0 refernce on valve.
        port1position = bytes(port_1_pos,'UTF-8')       #Creates the output and encodes it
        ser.write(port1position)                        #Sends encoded command to the Chem-on Valve
        time.sleep(t1)                                  #Wait for n seconds (based on end-user needs)

def port_2():
        print("Moving to port 2")

        port_2_pos ='AMA '+ str(port_p2)+'\r\n'         #Uses port_2 variable. AMA is absolute position from 0 refernce on valve.
        port2position = bytes(port_2_pos,'UTF-8')       #Creates the output and encodes it
        ser.write(port2position)                        #Position of port 2 relative to absolute position at initiliziation
        time.sleep(t1)                                  #Wait for n seconds

def port_3():
        print("Moving to port 3")

        port_3_pos ='AMA '+ str(port_p3)+'\r\n'         #Uses port_3 variable. AMA is absolute position from 0 refernce on valve.
        port3position = bytes(port_3_pos,'UTF-8')       #Creates the output and encodes it
        ser.write(port3position)                        #Position of port 3 relative to absolute position at initiliziation
        time.sleep(t1)                                  #Wait for n seconds

def port_4():
        print("Moving to port 4")

        port_4_pos ='AMA '+ str(port_p4)+'\r\n'         #Uses port_4 variable. AMA is absolute position from 0 refernce on valve.
        port4position = bytes(port_4_pos,'UTF-8')       #Creates the output and encodes it
        ser.write(port4position)                        #Position of port 4 relative to absolute position at initiliziation
        time.sleep(t1)                                  #Wait for n seconds

def port_5():
        print("Moving to port 5")

        port_5_pos ='AMA '+ str(port_p5)+'\r\n'         #Uses port_5 variable. AMA is absolute position from 0 refernce on valve.
        port5position = bytes(port_5_pos,'UTF-8')       #Creates the output and encodes it
        ser.write(port5position)                        #Position of port 5 relative to absolute position at initiliziation
        time.sleep(t1)                                  #Wait for n seconds

def port_6():
        print("Moving to port 6")
        port_6_pos ='AMA '+ str(port_p6)+'\r\n'         #Uses port_6 variable. AMA is absolute position from 0 refernce on valve.
        port6position = bytes(port_6_pos,'UTF-8')       #Creates the output and encodes it
        ser.write(port6position)                        #Position of port 6 relative to absolute position at initiliziation
        time.sleep(t1)                                  #Wait for n seconds
```

Appendix C3: MilliGat Pump Control

```
                                #DISPENSE/ASPIRATE FUNCTIONS

def blank_sample():
        print("Dispensing Blank Sample")

        # This section encodes the parameters set from PUMP VARIABLES section for Blank pump 1 speed and amount

        p1_blank_speed ='CVM '+ str(blank_p1_speed)+'*EU\r\n'
        blankp1speed = bytes(p1_blank_speed,'UTF-8')
        p1_blank_amount = 'CMR '+ str(blank_p1_amount)+'*EU\r\n'
        blankp1amount = bytes(p1_blank_amount,'UTF-8')

        ser.write(blankp1speed)                     #Pump 1 moves at speed set by blank_p1_speed variable
        ser.write(blankp1amount)                    #Pump 1 dispenses amount set by blank_p1_amount varaible
        time.sleep(blanktime)                       #wait for n seconds

def molybdate_reagent():
        print("Dispensing 320 ul pump_1/ aspirating 400 ul pump_2")

        # This section encodes the parameters set from PUMP VARIABLES section for Molybdate pump 1 speed and amount

        p1_molybdate_speed ='CVM '+ str(molybdate_p1_speed)+'*EU\r\n'
        molybdatep1speed = bytes(p1_molybdate_speed,'UTF-8')
        p1_molybdate_amount = 'CMR '+ str(molybdate_p1_amount)+'*EU\r\n'
        molybdatep1amount = bytes(p1_molybdate_amount,'UTF-8')

        # This section encodes the parameters set from PUMP VARIABLES section for Molybdate pump 2 speed and amount

        p2_molybdate_speed ='DVM '+ str(molybdate_p2_speed)+'*EU\r\n'
        molybdatep2speed = bytes(p2_molybdate_speed,'UTF-8')
        p2_molybdate_amount = 'DMR '+ str(molybdate_p2_amount)+'*EU\r\n'
        molybdatep2amount = bytes(p2_molybdate_amount,'UTF-8')

        ser.write(molybdatep1speed)                 #Pump 1 moves at speed set by molybdate_p1_speed variable
        ser.write(molybdatep1amount)                #Pump 1 dispenses amount set by molybdate_p1_amount varaible
        time.sleep(t0)
        ser.write(molybdatep2speed)                 #Pump 2 moves at speed set by molybdate_p2_speed variable
        ser.write(molybdatep2amount)                #Pump 2 aspirates amount set by molybdate_p2_amount varaible
        time.sleep(molybdatetime)                   #Wait for n seconds

def ascorbic_acid_reagent():
        print("Aspirating 400 ul pump_1/ Dispensing 320 ul pump_2")

        #This section encodes the parameters set from PUMP VARIABLES section for Ascorbic Acid pump 1 speed and amount

        p1_ascorbic_speed ='CVM '+ str(ascorbic_p1_speed)+'*EU\r\n'
        ascorbicp1speed = bytes(p1_ascorbic_speed,'UTF-8')
        p1_ascorbic_amount = 'CMR '+ str(ascorbic_p1_amount)+'*EU\r\n'
        ascorbicp1amount = bytes(p1_ascorbic_amount,'UTF-8')

        # This section encodes the parameters set from PUMP VARIABLES section for Ascorbic Acid pump 2 speed and amount

        p2_ascorbic_speed ='DVM '+ str(ascorbic_p2_speed)+'*EU\r\n'
        ascorbicp2speed = bytes(p2_ascorbic_speed,'UTF-8')
        p2_ascorbic_amount = 'DMR '+ str(ascorbic_p2_amount)+'*EU\r\n'
        ascorbicp2amount = bytes(p2_ascorbic_amount,'UTF-8')


        ser.write(ascorbicp1speed)                  #Pump 1 moves at speed set by ascorbic_p1_speed variable
        ser.write(ascorbicp1amount)                 #Pump 1 aspirates amount set by ascorbic_p1_amount varaible
        time.sleep(t0)
        ser.write(ascorbicp2speed)                  #Pump 2 moves at speed set by ascorbic_p2_speed variable
        ser.write(ascorbicp2amount)                 #Pump 2 aspirates amount set by ascorbic_p2_amount varaible
        time.sleep(ascorbicacidtime)                #Wait for n seconds
```

```
def flow_cell():
        print("Dispensing 400 ul pump_1 into flow cell")

        # This section encodes the parameters set from PUMP VARIABLES section for Flow Cell pump 1 speed and amount

        p1_flow_cell_speed ='CVM '+ str(flow_cell_p1_speed)+'*EU\r\n'
        flowcellp1speed = bytes(p1_flow_cell_speed,'UTF-8')
        p1_flow_cell_amount = 'CMR '+ str(flow_cell_p1_amount)+'*EU\r\n'
        flowcellp1amount = bytes(p1_flow_cell_amount,'UTF-8')


        ser.write(flowcellp1speed)                      #Pump 1 moves at speed set by flow_cell_p1_speed variable
        ser.write(flowcellp1amount)                     #Pump 1 dispenses amount set by flow_cell_p1_amount varaible
        time.sleep(flowcelltime)                        #Wait for n seconds

def po4_standard():
        print("Aspirating 600 ul pump_1 ")

        # This section encodes the parameters set from PUMP VARIABLES section for Po4 Standard pump 1 speed and amount

        p1_po4_standard_speed ='CVM '+ str(po4_standard_p1_speed)+'*EU\r\n'
        po4standardp1speed = bytes(p1_po4_standard_speed,'UTF-8')
        p1_po4_standard_amount = 'CMR '+ str(po4_standard_p1_amount)+'*EU\r\n'
        po4standardp1amount = bytes(p1_po4_standard_amount,'UTF-8')

        ser.write(po4standardp1speed)                   #Pump 1 moves at speed set by po4_standard_p1_speed variable
        ser.write(po4standardp1amount)                  #Pump 1 aspirates amount set by po4_standard_p1_amount varaible
        time.sleep(po4standardtime)                     #Wait for n seconds

def po4_sample():
        print("Aspirating 600 ul pump_1 ")

        # This section encodes the parameters set from PUMP VARIABLES section for Po4 Sample pump 1 speed and amount

        p1_po4_sample_speed ='CVM '+ str(po4_sample_p1_speed)+'*EU\r\n'
        po4samplep1speed = bytes(p1_po4_sample_speed,'UTF-8')
        p1_po4_sample_amount = 'CMR '+ str(po4_sample_p1_amount)+'*EU\r\n'
        po4samplep1amount = bytes(p1_po4_sample_amount,'UTF-8')

        ser.write(po4samplep1speed)                     #Pump 1 moves at speed set by po4_sample_p1_speed variable
        ser.write(po4samplep1amount)                    #Pump 1 aspirates amount set by po4_sample_p1_amount varaible
        time.sleep(po4sampletime)                       #Wait for n seconds
```

```
                                #SYSTEM FLUSH FUNCTION
def system_flush():
        print("System Flush In Progress")

        # This section encodes the parameters set from PUMP VARIABLES section for Flush pump 1 speed and amount

        p1_flush_speed ='CVM '+ str(flush_p1_speed)+'*EU\r\n'
        flushp1speed = bytes(p1_flush_speed,'UTF-8')
        p1_flush_amount = 'CMR '+ str(flush_p1_amount)+'*EU\r\n'
        flushp1amount = bytes(p1_flush_amount,'UTF-8')

        # This section encodes the parameters set from PUMP VARIABLES section for Flush pump 2 speed and amount

        p2_flush_speed ='DVM '+ str(flush_p2_speed)+'*EU\r\n'
        flushp2speed = bytes(p2_flush_speed,'UTF-8')
        p2_flush_amount = 'DMR '+ str(flush_p2_amount)+'*EU\r\n'
        flushp2amount = bytes(p2_flush_amount,'UTF-8')


        ser.write(flushp1speed)                         #Pump 1 moves at speed set by flush_p1_speed variable
        ser.write(flushp1amount)                        #Pump 1 dispenses amount set by flush_p1_amount varaible
        time.sleep(t0)
        ser.write(flushp2speed)                         #Pump 2 moves at speed set by flush_p2_speed variable
        ser.write(flushp2amount)                        #Pump 2 dispenses amount set by flush_p2_amount varaible
        time.sleep(flushtime)
```

Appendix C4: Seabreeze Spectrometer Control

```
                                        #SPECTOMETER FUNCTIONS
def spectro_darkscan(spec):
        #place code for darkscan
        '''
        Inputs:
        -Spectrophotometer read from Seabreeze Library

        Outputs:
        -Intensity values across all wavelengths taken with lamp off.
        '''

        time.sleep(10) #sleep 10s to ensure no light transmission after lamp turned off

        wavelengths = spec.wavelengths()
        intensities = spec.intensities()
        column_names = ['wavelengths','intensities']
        combine = np.vstack((wavelengths, intensities)).T
        np.shape(combine)

        darkscan = pd.DataFrame(data=combine, columns=column_names) #produces dataframe from most recent scan taken from spec.

        dark_spec = darkscan['intensities']

        plt.figure()
        plt.xlabel('Wavelength nm')
        plt.ylabel('Intensity')
        plt.title('Darkscan Calibration')
        plt.plot(wavelengths,dark_spec)
        plt.show()
        time.sleep(5)

        return dark_spec

def spectro_refscan(spec):
        #place code for reference scan
        '''
        Inputs:
        -Spectrophotometer read from Seabreeze Library

        Outputs:
        -Intensity values across all wavelengths taken with lamp on, no sample in flow cell.
        '''

        refscan_intensity = []

        samples_to_average = 20

        for i in range(samples_to_average):

            wavelengths = spec.wavelengths()
            intensities = spec.intensities()
            column_names = ['wavelengths','intensities']
            combine = np.vstack((wavelengths, intensities)).T
            np.shape(combine)
            refscan = pd.DataFrame(data=combine, columns=column_names) #produces dataframe from most recent scan taken from spec.

            refscan_intensity.append(refscan['intensities'])
            time.sleep(0.25)

        ref_spec = np.array(refscan_intensity).mean(axis=0)

        plt.figure()
        plt.xlabel('Wavelength nm')
        plt.ylabel('Intensity')
        plt.title('Reference Scan')
        plt.plot(wavelengths,intensities)
        plt.show()
        time.sleep(5)
        return ref_spec
```

```
def spectro_samplescan(spec):
        #place code for sample scans
        '''
        Inputs:
        -Spectrophotometer read from Seabreeze Library

        Outputs:
        -Intensity values across all wavelengths taken with lamp on, sample in flow cell.
        '''

        sampscan_intensity = []

        samples_to_average = 20

        for i in range(samples_to_average):

            wavelengths = spec.wavelengths()
            intensities = spec.intensities()
            column_names = ['wavelengths','intensities']
            combine = np.vstack((wavelengths, intensities)).T
            np.shape(combine)
            sampscan = pd.DataFrame(data=combine, columns=column_names) #produces dataframe from most recent scan taken from spec.

            sampscan_intensity.append(sampscan['intensities'])

            time.sleep(0.25)

        samp_spec = np.array(sampscan_intensity).mean(axis=0)

        samp_lambdas = sampscan['wavelengths']

        plt.figure()
        plt.xlabel('Wavelength nm')
        plt.ylabel('Intensity')
        plt.title('Sample Scan')
        plt.plot(wavelengths,intensities)
        plt.show()
        time.sleep(5)

        return samp_spec, samp_lambdas

def spectro_calcAbsorbance(dark_spec,ref_spec,samp_spec,samp_lambdas):

        '''
        This function uses outputs from the following functions for its input: "darkscan", "referencescan", "samplescan". Note that these functions must be run prior to running this function.

        The inputs are defined are follows:
        dark_spec = darkscan(spec)
        ref_spec = referencescan(spec)
        samp_spec, samp_lambdas = samplescan(spec)[:]

        Absorbance is calculated using the equation: A = log_10 (Io - dark signal / I - dark signal), where Io is the intensity from the reference scan, and I is the intensity from the sample scan.

        Outputs:
        -Final absorbance value to be used in calculating concentration of analyte.
        -Absorbance spectrum across all available wavelengths
        '''

        absorbances_unfiltered = np.log10((ref_spec - dark_spec) / (samp_spec - dark_spec))     #absorbance ranges 0 to 1

        ##smoothing spectrum using moving average
        window_size = 20
        numbers_series = pd.Series(absorbances_unfiltered)
        windows = numbers_series.rolling(window_size)
        moving_averages = windows.mean()
        moving_averages_list = moving_averages.tolist()
        absorbances_filtered = moving_averages_list[window_size - 1:]
        wavelengths_filtered = samp_lambdas[:-19]

        #organizing filtered outputs into dataframe
        column_names = ['wavelengths_filtered','absorbances_filtered']
        combine = np.vstack((wavelengths_filtered, absorbances_filtered)).T
        absorbances_final = pd.DataFrame(data=combine, columns=column_names)

        #calculating absorbance value at monitoring wavelength
        abs_monitoring = absorbances_final[(absorbances_final['wavelengths_filtered'] < (monitoringlambda + 1)) & (absorbances_final['wavelengths_filtered'] > (monitoringlambda - 1))]
        abs_monitoring_mean = np.mean(abs_monitoring['absorbances_filtered'])

        ##calculating absorbance value at reference wavelength
        abs_reference = absorbances_final[(absorbances_final['wavelengths_filtered'] < (reflambda + 1)) & (absorbances_final['wavelengths_filtered'] > (reflambda - 1))]
        abs_reference_mean = np.mean(abs_reference['absorbances_filtered'])

        abs_final = abs_monitoring_mean - abs_reference_mean
        print('abs_final')
        print(abs_final)
```

Appendix C5: Pump Priming

```
def port_1prime():
        port_1()
        port1_primep1speed ='CVM '+ str(port1_pp1speed)+'*EU\r\n'
        port1primep1speed = bytes(port1_primep1speed,'UTF-8')
        port_1primep1amount = 'CMR '+ str(port1_pp1amount)+'*EU\r\n'
        port1primep1amount = bytes(port_1primep1amount,'UTF-8')
        ser.write(port1primep1speed)
        ser.write(port1primep1amount)
        time.sleep(p1primetime)

def port_2prime():
        port_2()
        port2_primep1speed ='CVM '+ str(port2_pp1speed)+'*EU\r\n'
        port2primep1speed = bytes(port2_primep1speed,'UTF-8')
        port_2primep1amount = 'CMR '+ str(port2_pp1amount)+'*EU\r\n'
        port2primep1amount = bytes(port_2primep1amount,'UTF-8')

        port2_primep2speed ='DVM '+ str(port2_pp2speed)+'*EU\r\n'
        port2primep2speed = bytes(port2_primep2speed,'UTF-8')
        port_2primep2amount = 'DMR '+ str(port2_pp2amount)+'*EU\r\n'
        port2primep2amount = bytes(port_2primep2amount,'UTF-8')

        ser.write(port2primep1speed)
        ser.write(port2primep1amount)
        time.sleep(t0)
        ser.write(port2primep2speed)
        ser.write(port2primep2amount)
        time.sleep(p2primetime)

def port_3prime():
        port_3()
        port3_primep1speed ='CVM '+ str(port3_pp1speed)+'*EU\r\n'
        port3primep1speed = bytes(port3_primep1speed,'UTF-8')
        port_3primep1amount = 'CMR '+ str(port3_pp1amount)+'*EU\r\n'
        port3primep1amount = bytes(port_3primep1amount,'UTF-8')
        ser.write(port3primep1speed)
        ser.write(port3primep1amount)
        time.sleep(p3primetime)

def port_4prime():
        port_4()
        port4_primep1speed ='CVM '+ str(port4_pp1speed)+'*EU\r\n'
        port4primep1speed = bytes(port4_primep1speed,'UTF-8')
        port_4primep1amount = 'CMR '+ str(port4_pp1amount)+'*EU\r\n'
        port4primep1amount = bytes(port_4primep1amount,'UTF-8')
        ser.write(port4primep1speed)
        ser.write(port4primep1amount)
        time.sleep(p4primetime)

def port_5prime():
        port_5()
        port5_primep1speed ='CVM '+ str(port5_pp1speed)+'*EU\r\n'
        port5primep1speed = bytes(port5_primep1speed,'UTF-8')
        port_5primep1amount = 'CMR '+ str(port5_pp1amount)+'*EU\r\n'
        port5primep1amount = bytes(port_5primep1amount,'UTF-8')
        ser.write(port5primep1speed)
        ser.write(port5primep1amount)
        time.sleep(p5primetime)

def port_6prime():
        port_6()
        port6_primep1speed ='CVM '+ str(port6_pp1speed)+'*EU\r\n'
        port6primep1speed = bytes(port6_primep1speed,'UTF-8')
        port_6primep1amount = 'CMR '+ str(port6_pp1amount)+'*EU\r\n'
        port6primep1amount = bytes(port_6primep1amount,'UTF-8')
        ser.write(port6primep1speed)
        ser.write(port6primep1amount)
        time.sleep(p6primetime)

def prime():
        for i in range(totalprimes):
                port_1()
                port_2prime()
                port_1prime()
                port_3prime()
                port_1prime()
                port_4prime()
                port_1prime()
                port_5prime()
                port_1prime()
                port_6prime()
                port_1prime()
                auxMotor()
        print('Prime Complete')
```

Appendix C6: Moss Landing Sequence Control

```
                                        #BLANK/STANDARD/SAMPLE RUN FUNCTIONS
def blank_sample_run():
        port_1()
        port_2()
        system_flush()
        lightOff()
        time.sleep(darkscantime)
        dark_spec=spectro_darkscan(spec)
        lightOn()
        time.sleep(refscantime)
        ref_spec=spectro_refscan(spec)

        blank_sample()
        port_3()
        molybdate_reagent()
        port_5()
        ascorbic_acid_reagent()
        port_2()
        flow_cell()
        samp_spec, samp_lambdas = spectro_samplescan(spec)[:]
        absorbance_blank=spectro_calcAbsorbance(dark_spec,ref_spec,samp_spec, samp_lambdas)
        return absorbance_blank

def po4_standard_run():
        port_1()
        port_2()
        system_flush()
        lightOff()
        time.sleep(darkscantime)
        dark_spec=spectro_darkscan(spec)
        lightOn()
        time.sleep(refscantime)
        ref_spec=spectro_refscan(spec)


        port_6()
        po4_standard()
        port_3()
        molybdate_reagent()
        port_5()
        ascorbic_acid_reagent()
        port_2()
        flow_cell()
        samp_spec, samp_lambdas = spectro_samplescan(spec)[:]
        absorbance_po4std=spectro_calcAbsorbance(dark_spec,ref_spec,samp_spec, samp_lambdas)
        return absorbance_po4std

def po4_sample_run():
        port_1()
        port_2()
        system_flush()
        lightOff()
        time.sleep(darkscantime)
        dark_spec=spectro_darkscan(spec)
        lightOn()
        time.sleep(refscantime)
        ref_spec=spectro_refscan(spec)


        auxMotor()
        port_4()
        po4_sample()
        port_3()
        molybdate_reagent()
        port_5()
        ascorbic_acid_reagent()
        port_2()
        flow_cell()
        samp_spec, samp_lambdas = spectro_samplescan(spec)[:]
        absorbance_po4samp=spectro_calcAbsorbance(dark_spec,ref_spec,samp_spec, samp_lambdas)
        lightOff()
        return absorbance_po4samp

```

Appendix C7: Aux Motor And LED Lamp Control

```
578                                         #AUX MOTOR AND LIGHTSOURCE FUNCTIONS
579   def auxMotor():
580           print('Aux start')
581           GPIO.output(motorPin,GPIO.HIGH)
582           time.sleep(auxtime)
583           GPIO.output(motorPin,GPIO.LOW)
584           print('Aux stopped')
585           time.sleep(t1)
586
587   def lightOn():
588           GPIO.output(ledPin,GPIO.HIGH)
589           print('Light On')
590
591   def lightOff():
592           GPIO.output(ledPin,GPIO.LOW)
593           print('Light Off')
```

Appendix C8: Autonomous Control Cycle

```
669   #START OF CODE
670   #The system will run the blank sample sequence twice, then it will follow with the PO4 standard twice, then
671   #there will be one PO4 seawater sample run only three times. The system will then post process to calculate the
672   #the PO4 concentration. The system will then sleep for ~40 mins., then run only the PO4 seawater sample again, followed by
673   #and additional sleep cycle. This will be repeated again, then calibration process will be restarted(Running the complete
674   #code from the beginning)
675
676
677   blank = [0, 0]
678   stand = [0,0]
679   sample = [0,0,0]
680   po4conc_array=[]
681
682   while TRUE:
683
684           for i in range(2):
685                   blank[i]= blank_sample_run()
686
687           for i in range(2):
688                   stand[i] = po4_standard_run()
689
690           mean_abs_blank=(blank[0]+blank[1])/2
691           mean_abs_stand=(stand[0]+stand[1])/2
692
693           for i in range(3):
694                   sample[i]=po4_sample_run()
695                   po4conc=(sample(i)-mean_abs_blank)*(known_concentration/(mean_abs_stand-mean_abs_blank))
696                   po4conc_array.append(po4conc)
697
698
699   GPIO.cleanup()
700   ser.close()
```

Appendix C9: Insitu PO4 Sequence

	Sequence steps		Volumes/ flow rates	Notes
Activate instrument	Turn on analyzer			
	Turn light off			
	Get dark scan (see seabreeze code)			
	Turn lamp on			
	Wait 3 minutes for lamp to warm up			
Run Blank (2x)	Get reference scan (sea seabreeze code)			
	Pump 1 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	600,60,1	Dispensing blank sample (carrier)
	COV-6 go to port 3 (Molybdate reagent)			
	Pump 2 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	400,40,0	Pumps run simultaneously
	Pump 1 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	320,32,1	
	Wait 2 seconds			
	COV-6 go to port 5 (Ascorbic acid reagent)			
	Pump 1 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	400,40,0	Pumps run simultaneously

	Pump 2 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	320,32,1	
	Wait 2 seconds			
	Go to port 2 (flow cell)			
	Pump 1 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	400,25,1	***depends on flow cell length (used 300 with 10cm FC)
	Wait 5 minutes			**Could be shortened given no PO4 in blank
	Get absorbance value of blank sample (see seabreeze code)			
	Pump 1 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	1000,150,0	Pumps run simultaneouslty to flush instrument
	Pump 2 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	1000,150,1	
Run PO4 standards (2x)	Get reference scan (sea seabreeze code)			
	COV-6 go to port 6 (PO4 standard)			
	Pump 1 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	600,60,1	
	COV-6 go to port 3 (Molybdate reagent)			
	Pump 2 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	400,40,0	Pumps run simultaneously
	Pump 1 dispense	volume (µL), flow rate	320,32,1	

	(µL/s), hold all (0/1)		
Wait 2 seconds			
COV-6 go to port 5 (Ascorbic acid reagent)			
Pump 1 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	400,40,0	Pumps run simultaneously
Pump 2 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	320,32,1	
Wait 2 seconds			
Go to port 2 (flow cell)			
Pump 1 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	400,25,1	***depends on flow cell length (used 300 with 10cm FC)
Wait 5 minutes			
Get absorbance value of PO4 standard (see seabreeze code)			
COV-6 go to port 3 (Molybdate reagent)			Priming step: added 4/4/22
Pump 1 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	150,50,1	
COV-6 go to port 6 (PO4 standard)			
Pump 1 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	150,50,1	

	Go to port 2 (flow cell)			
	Pump 1 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	1000,150,0	Pumps run simultaneously to flush instrument
	Pump 2 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	1000,150,1	
Run PO4 seawater sample (1x)	Get reference scan (sea seabreeze code)			
	Run auxiliary pump for 60 seconds			
	COV-6 go to port 4 (PO4 sample)			
	Pump 1 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	600,60,1	
	COV-6 go to port 3 (Molybdate reagent)			
	Pump 2 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	400,40,0	Pumps run simultaneously
	Pump 1 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	320,32,1	
	Wait 2 seconds			
	COV-6 go to port 5 (Ascorbic acid reagent)			
	Pump 1 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	400,40,0	Pumps run simultaneously
	Pump 2 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	320,32,1	

	Wait 2 seconds			
	Go to port 2 (flow cell)			
	Pump 1 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	400,25,1	
	Wait 5 minutes			
	Get absorbance value of PO4 sample (see seabreeze code)			
	COV-6 go to port 3 (Molybdate reagent)			Priming step: added 4/4/22
	Pump 1 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	150,50,1	
	COV-6 go to port 4 (PO4 sample)			
	Pump 1 aspirate	volume (µL), flow rate (µL/s), hold all (0/1)	150,50,1	
	Go to port 2 (flow cell)			
	Pump 1 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	1000,150,0	Pumps run simultaneously to flush instrument
	Pump 2 dispense	volume (µL), flow rate (µL/s), hold all (0/1)	1000,150,1	
post-processing	Calculate PO4 concentration:			
	[PO4] = (Abs_sample - mean_Abs_blank) x [concentration of po4 standard] / (mean_Abs_standard-mean_Abs_blank)			

	Store sample time stamp + sample concentration, send to server			
	Sleep until next sample (~40 min if wanting hourly measurements)			

Appendix D: Datasheets

Appendix D1: GlobalFIA User Manual

https://www.globalfia.com/images/Documents/FloZF_5.0_manual.pdf

Appendix D2: GlobalFIA milliGAT Pump User Manual

https://www.globalfia.com/images/milliGAT_3.11.pdf

Appendix D3: GlobalFIA miniSIA-2 User Manual

https://www.globalfia.com/images/2015Brochures/man_miniSIA-2_-.pdf

Appendix D4: MDrive 17Plus User Manual

https://www.novantaims.com/downloads/manuals/MDI17_23_Plus.pdf

Appendix D5: MDrive 17Plus Software Reference Manual

https://www.servotechnica.ru/files/doc/documents/file-257.pd

Appendix D6: FTDI RS485 Converter Datasheet

https://ftdichip.com/wp-content/uploads/2020/08/DS_FT232R.pdf

Appendix D7: Virtual USB Port and Serial EEPROM interface Datasheet

https://www.ti.com/lit/ds/symlink/tusb2077a.pdf?ts=1652754519022&ref_url=https%253A%252F%252Fwww.google.com%252F

Appendix D9: Max485 Transceiver Datasheet

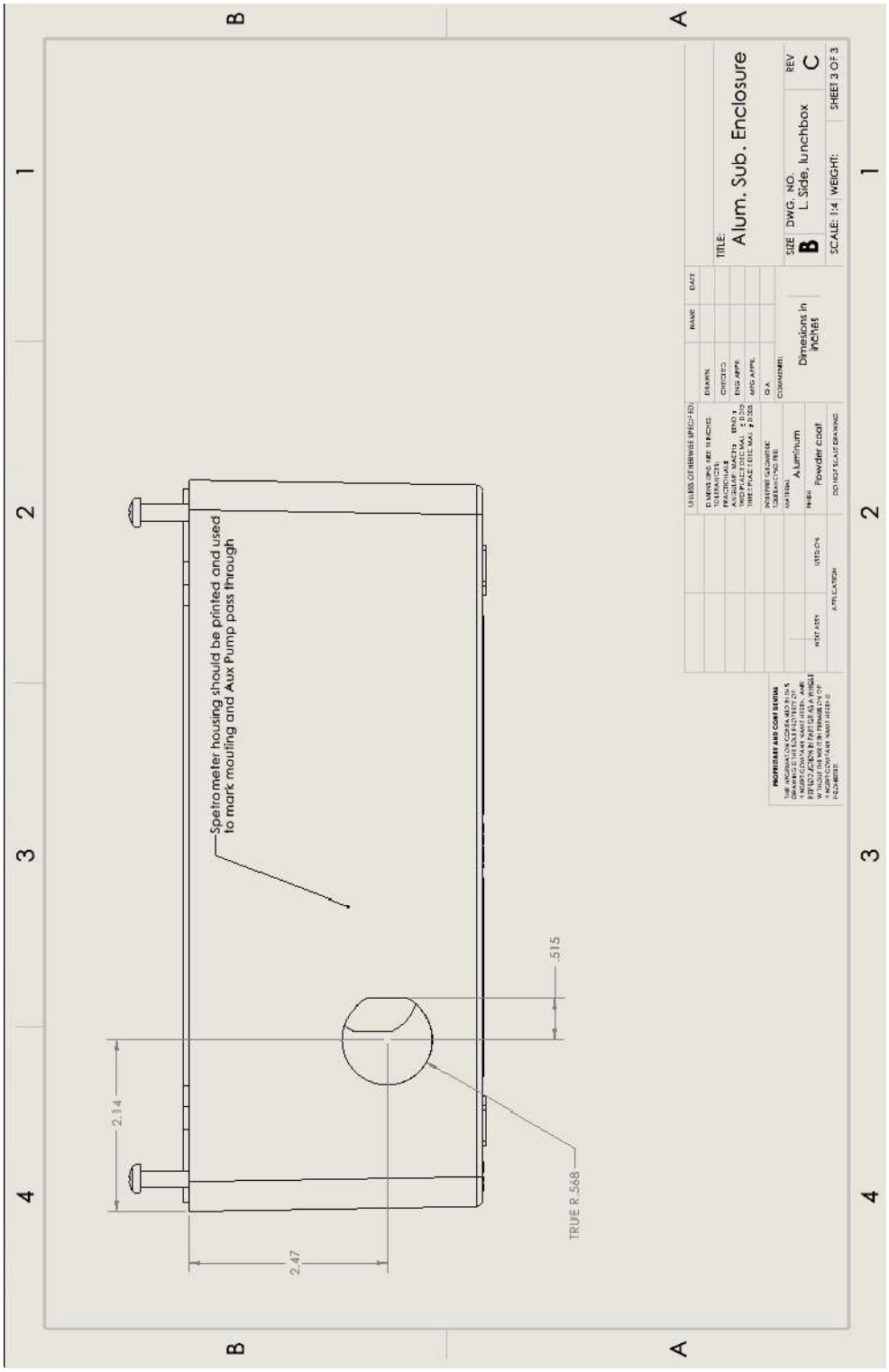
https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf

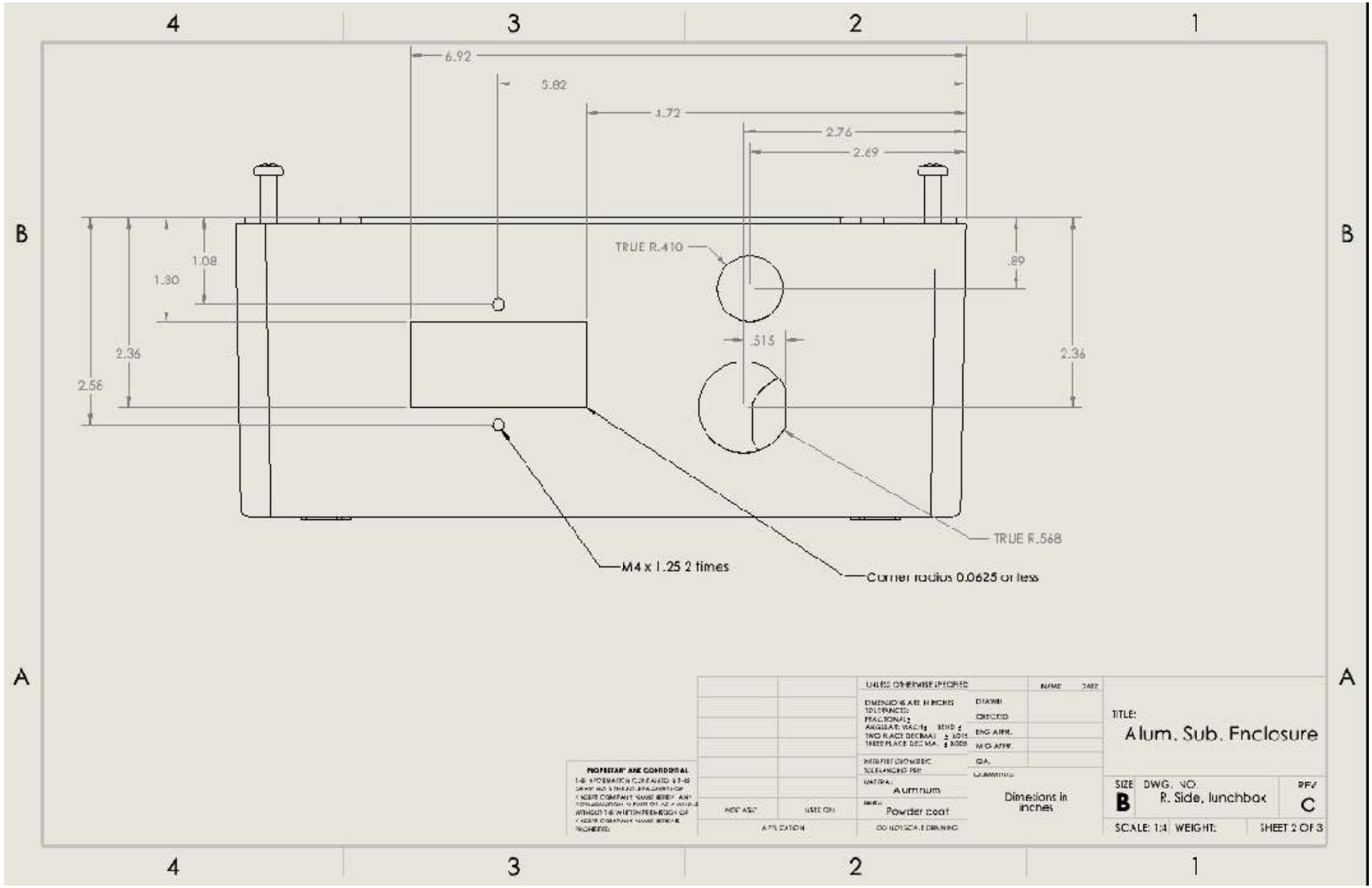
Appendix D10: Sequent Microsystems Industrial Interface PCB

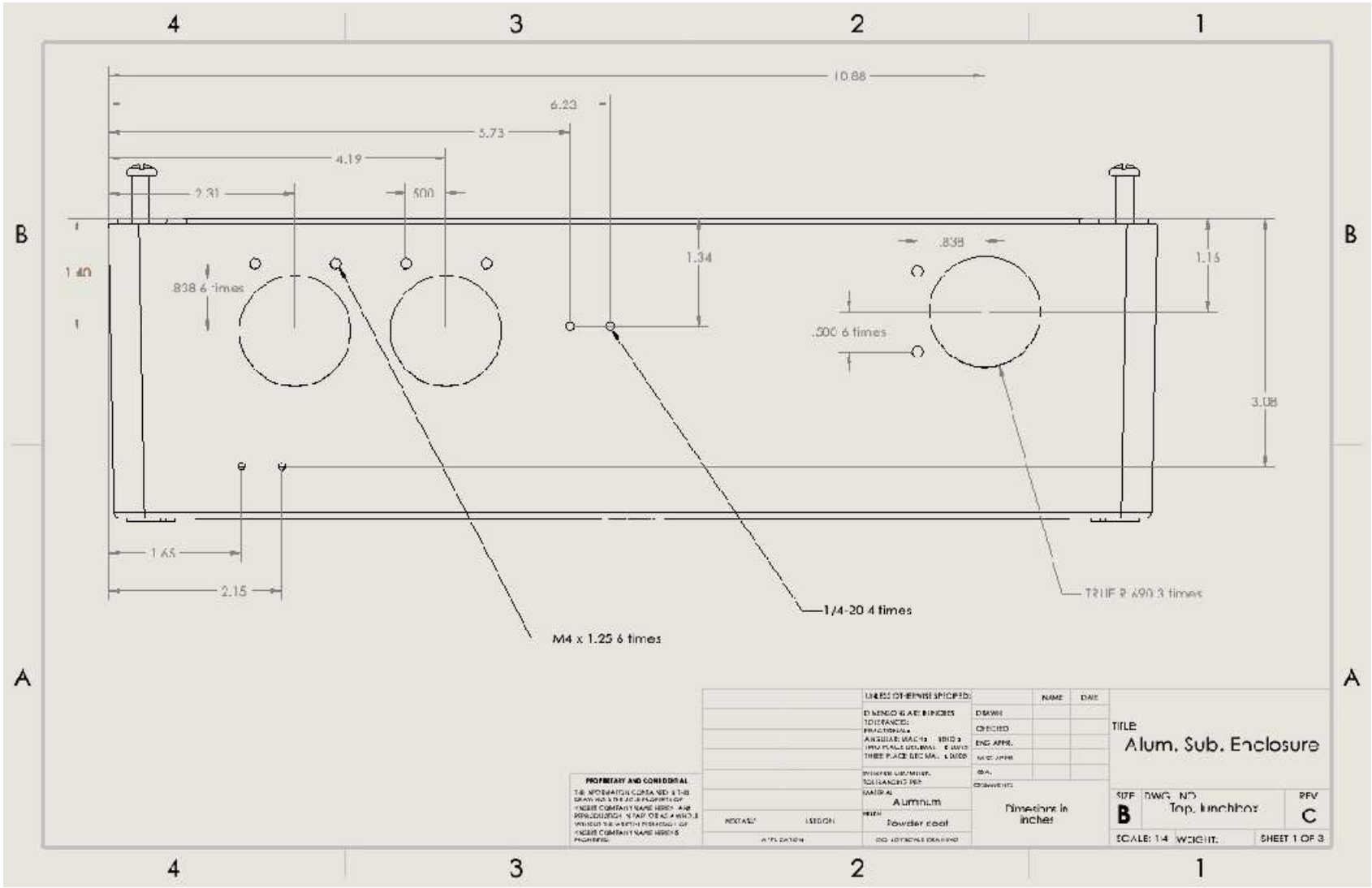
https://sequentmicrosystems.com/pages/industrial-automation-downloads

Appendix E: Component Drawings

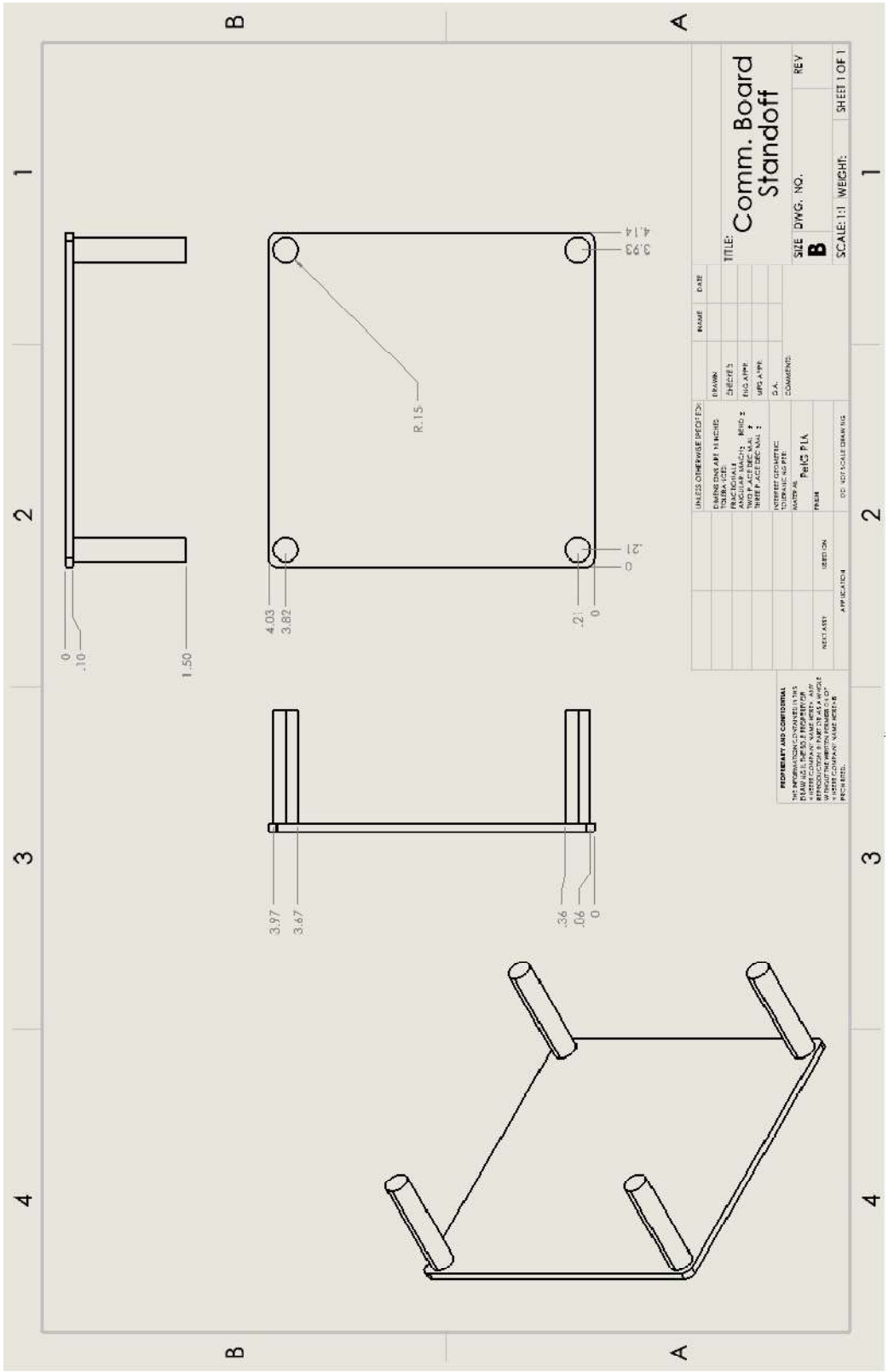
Appendix E1: Aluminum Sub. Enclosure



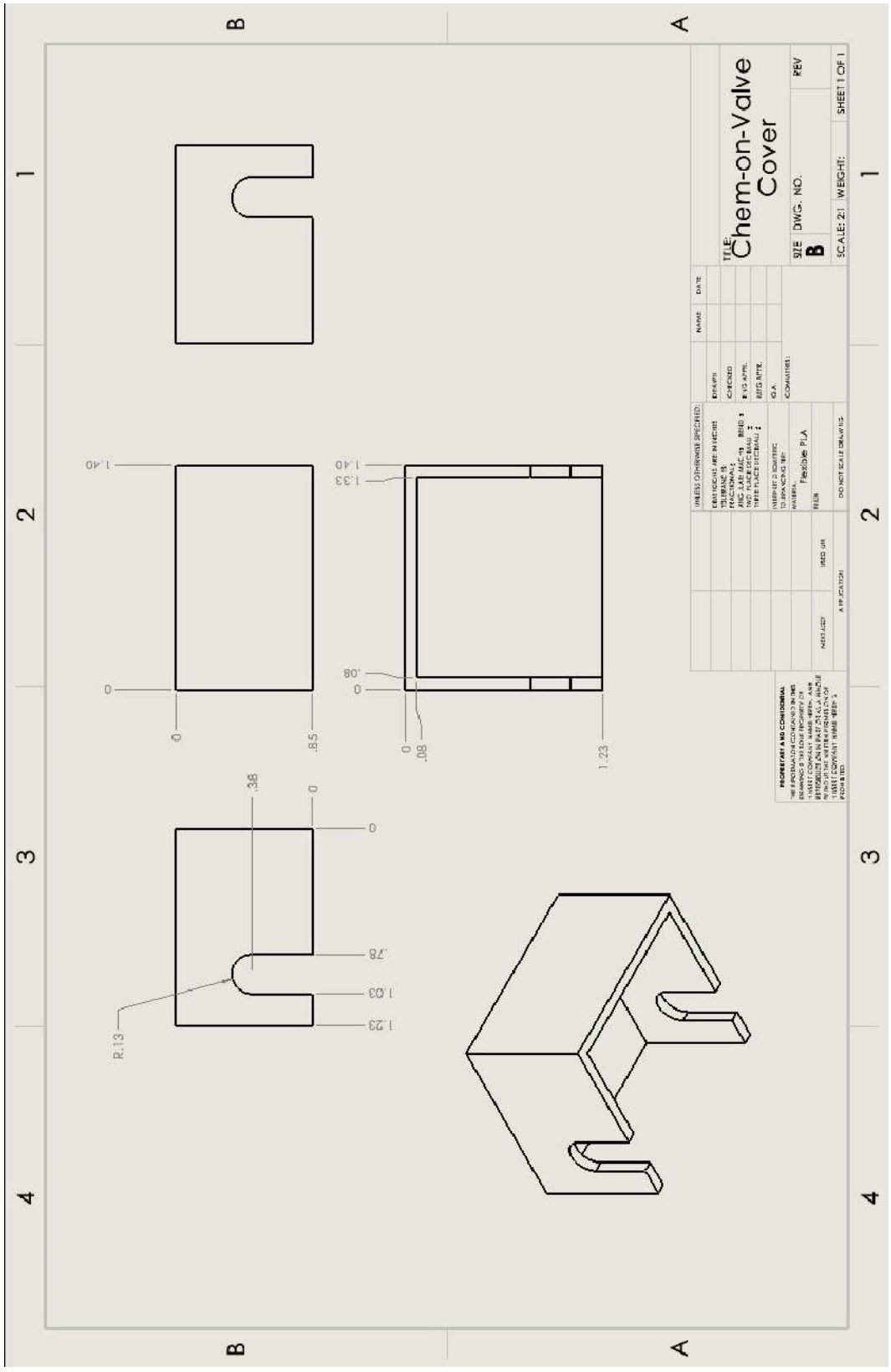

4
3
2
1
B
A
6.92
5.82
4.72
2.76
2.69
TRUE R.410
1.08
1.30
.89
.515
2.36
2.58
2.36
TRUE R.568
M4 x 1.25 2 times
Corner radius 0.0625 or less
UNLESS OTHERWISE SPECIFIED:
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
QA
COMMENTS:
MATERIAL
Aluminum
FINISH
Powder coat
Dimensions in inches
TITLE:
Alum. Sub. Enclosure
SIZE
B
DWG. NO.
R. Side, lunchbox
REV
C
SCALE: 1:4
WEIGHT:
SHEET 2 OF 3
PROPRIETARY AND CONFIDENTIAL
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING


10.88
6.23
5.73
4.19
2.31
.500
1.34
.838
1.16
1.40
.838 6 times
.500 6 times
3.08
1.65
2.15
M4 x 1.25 6 times
1/4-20 4 times
TRUE R .690 3 times
UNLESS OTHERWISE SPECIFIED:
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS
MATERIAL
Aluminum
FINISH
Powder coat
Dimensions in inches
TITLE
Alum. Sub. Enclosure
SIZE
B
DWG. NO.
Top_Linchbar
REV
C
SCALE: 1:4
WEIGHT:
SHEET 1 OF 3
PROPRIETARY AND CONFIDENTIAL

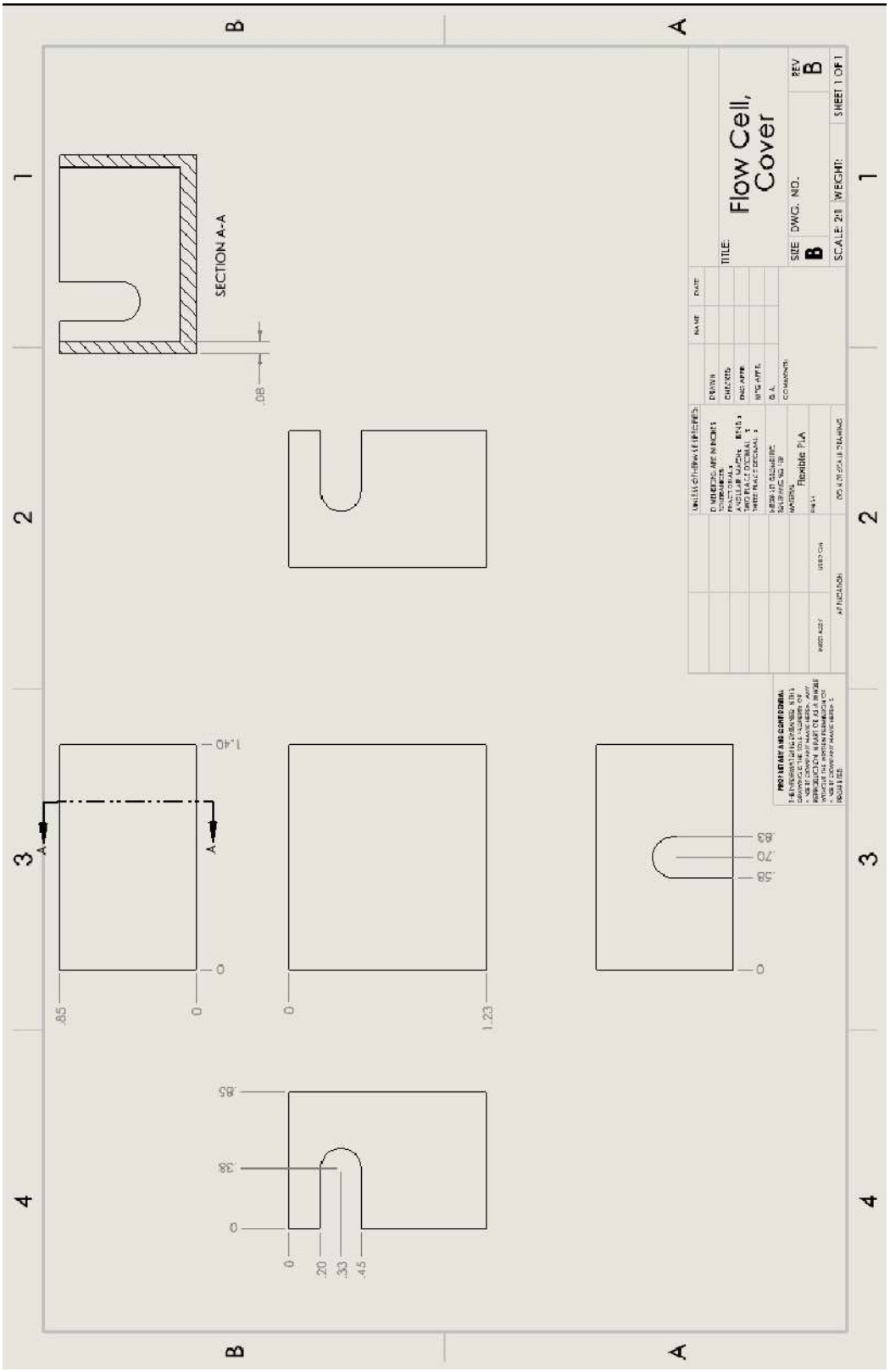
Appendix E2: Communications Board Standoff



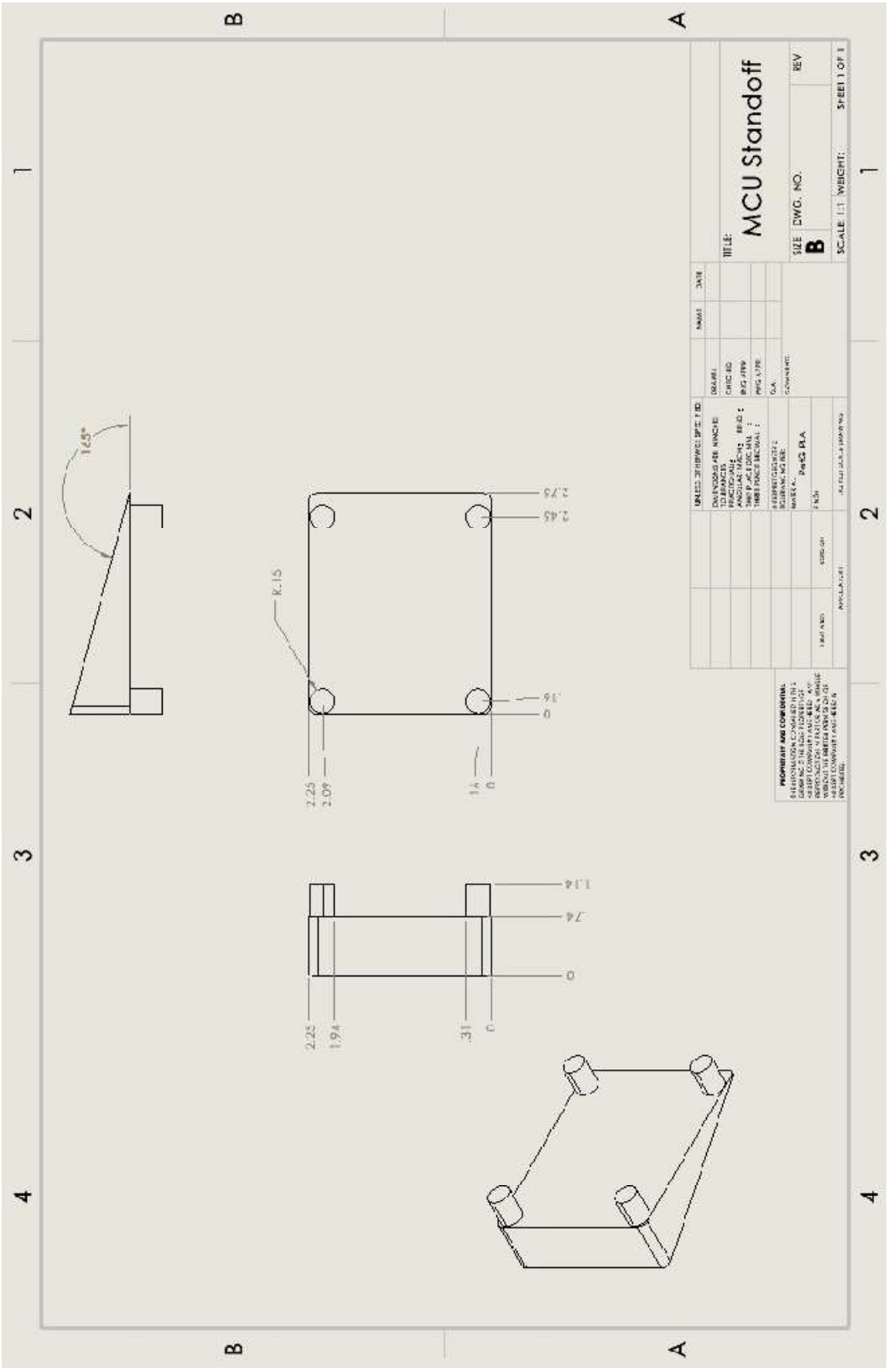
Appendix E3: Chem On Valve Cover



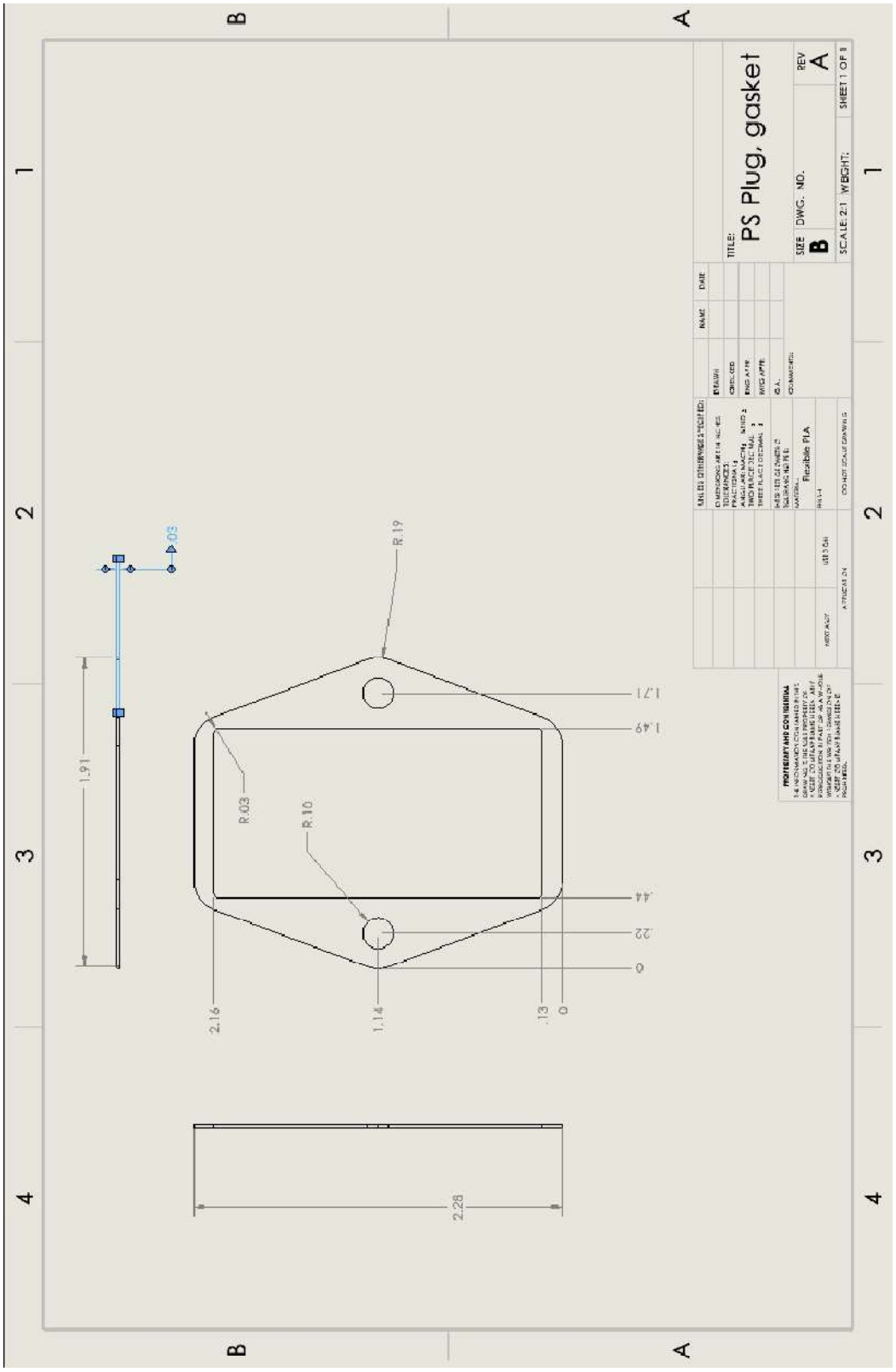
Appendix E4: Flow Cell Cover



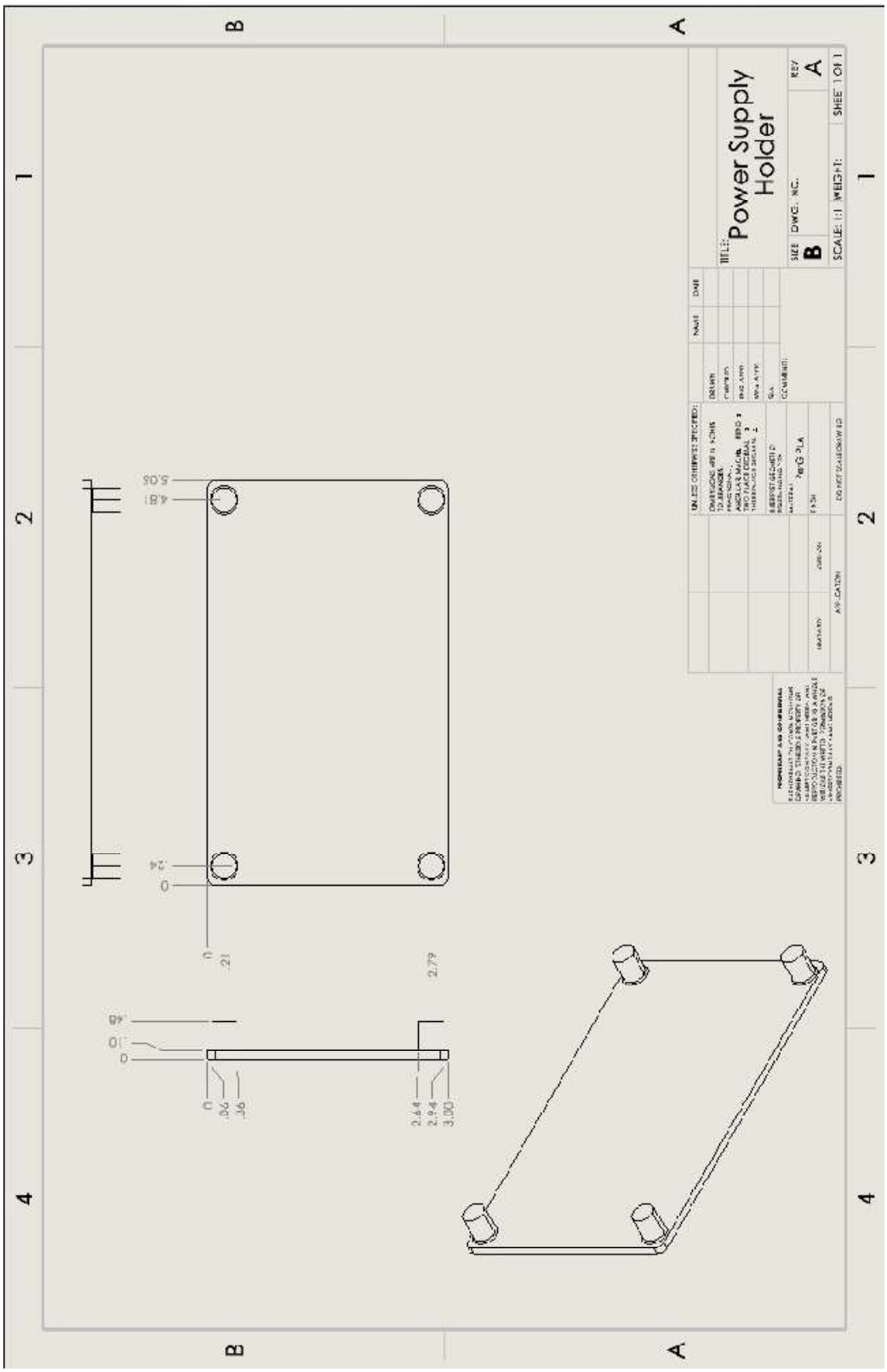
Appendix E5: Raspberry Pi4 Standoff



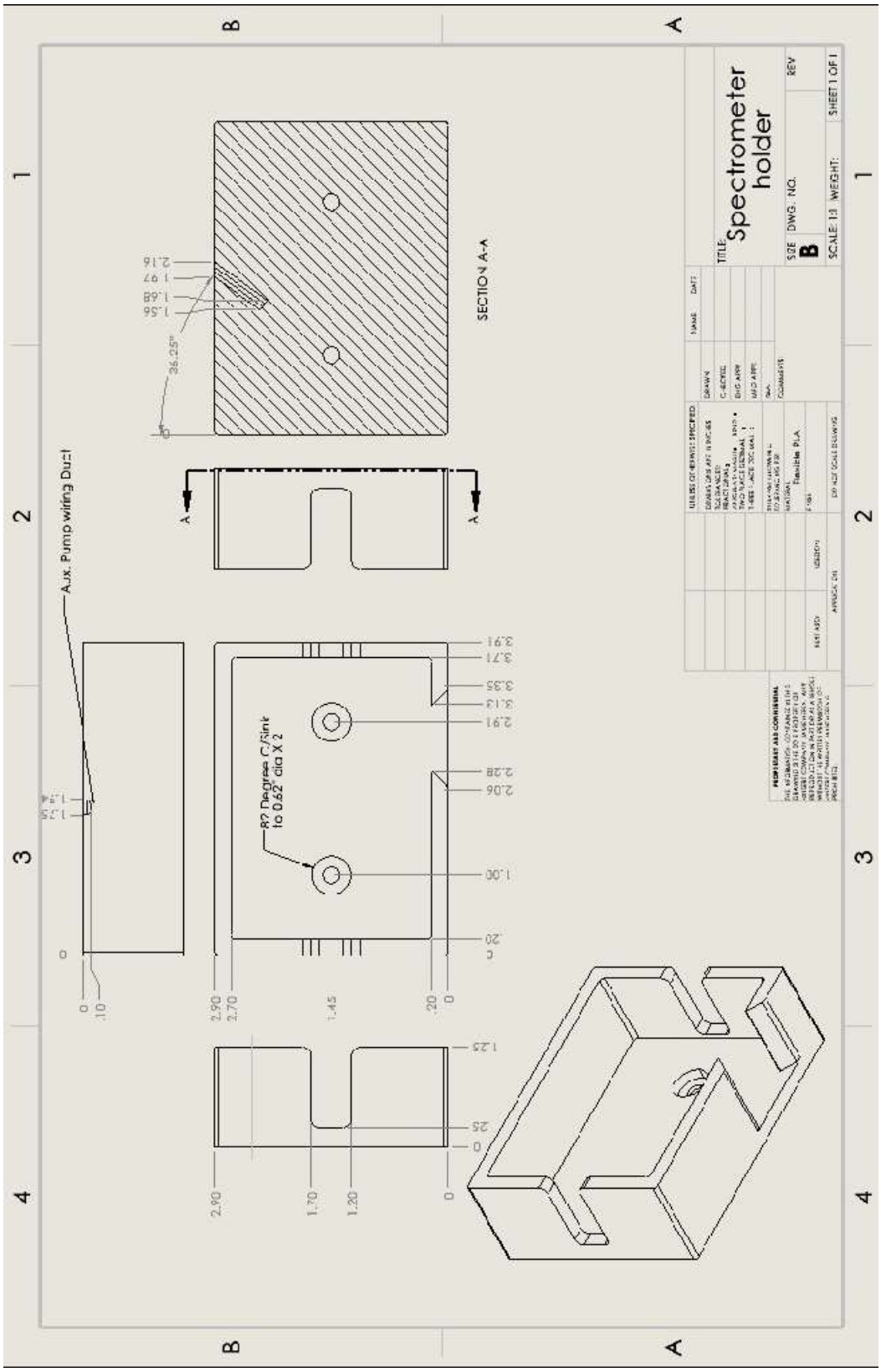
Appendix E6: Power Supply Gasket



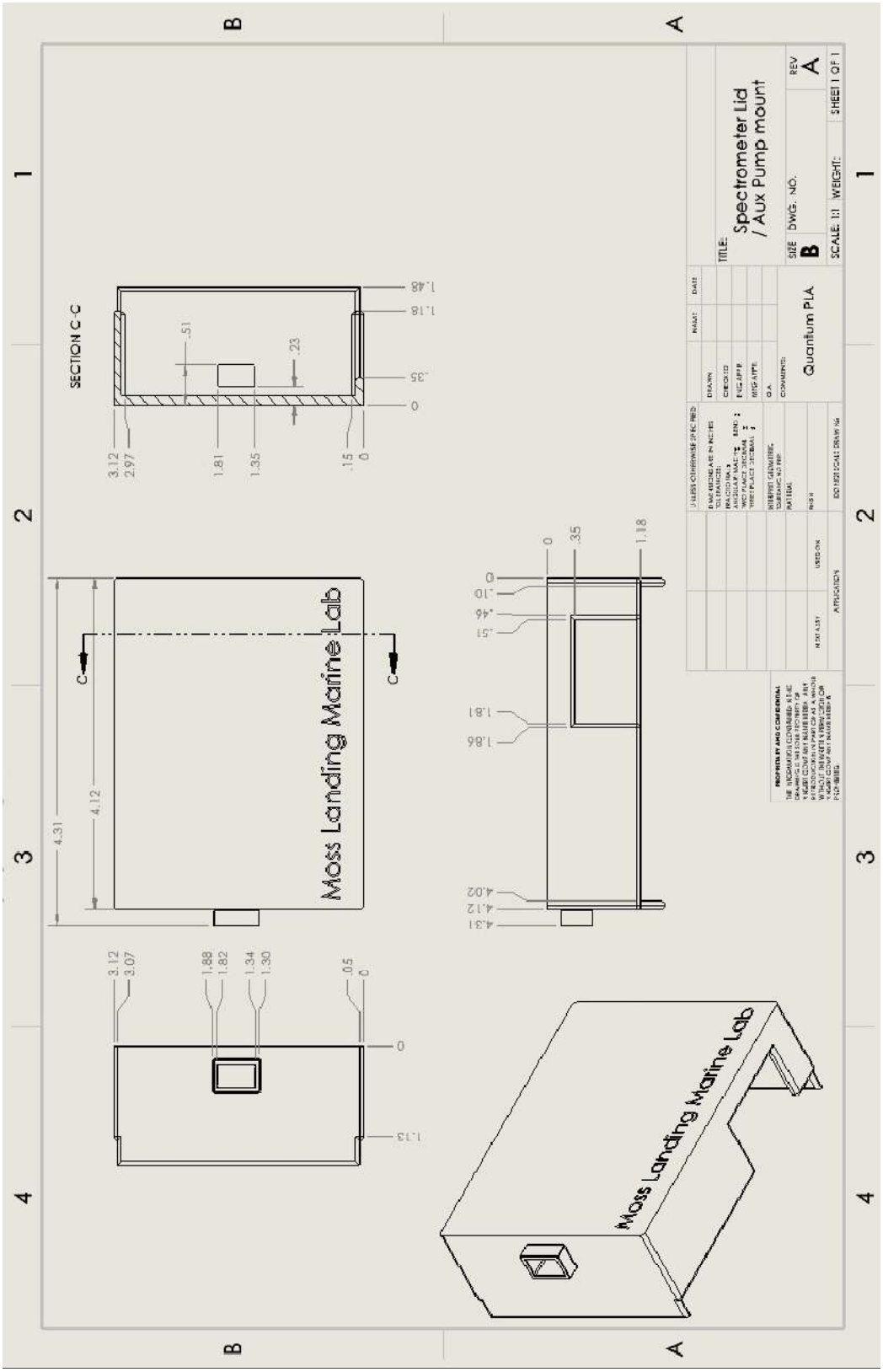
Appendix E7: Power Supply Holder



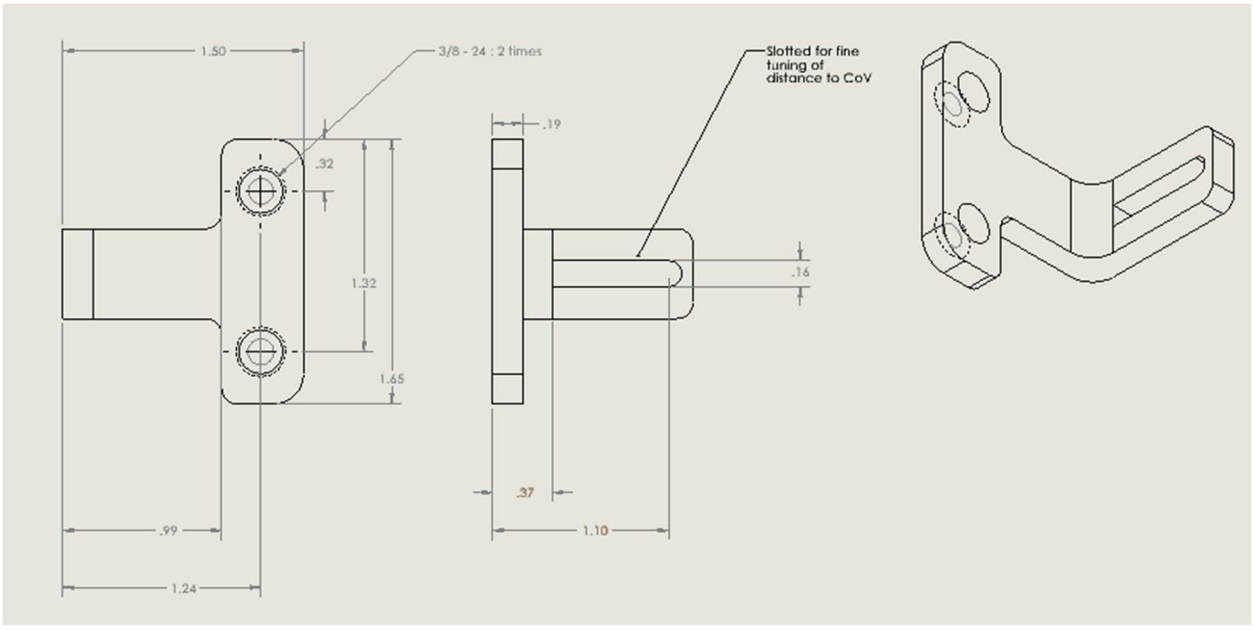
Appendix E8: Spectrometer Holder

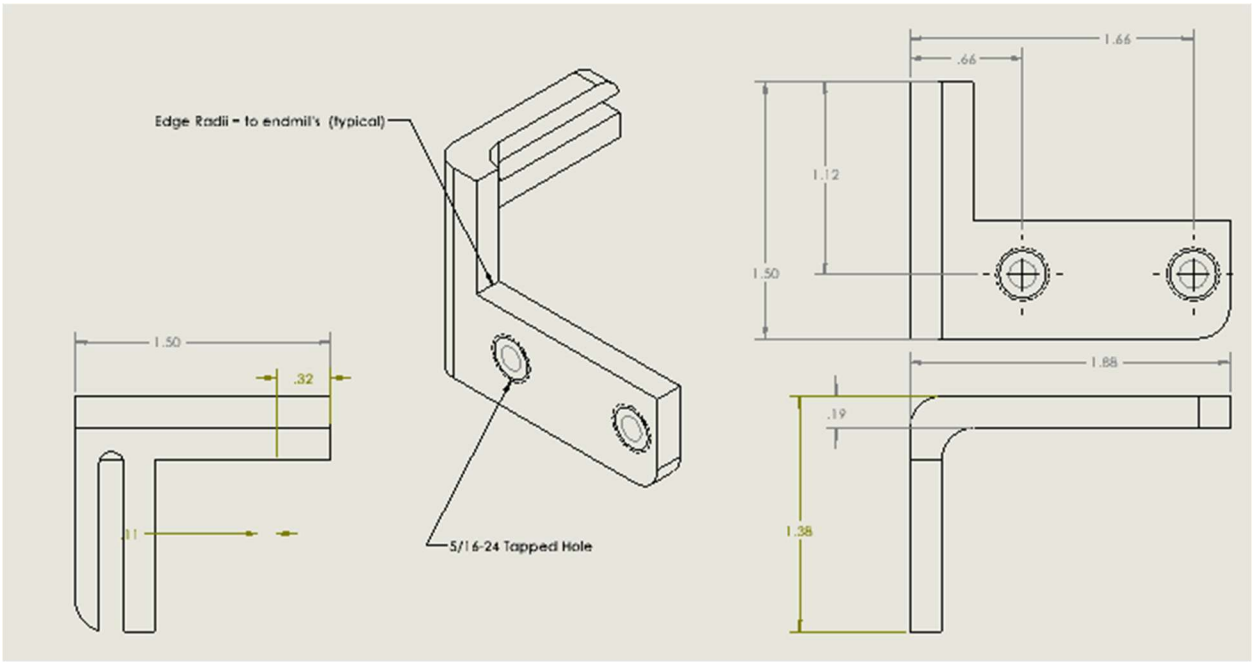


Appendix E9: Spectrometer Lid/ Aux Motor Mount



Appendix E10: Fabricated Holding Coil Mounts





Appendix F: Bill of Materials

Appendix F1: Component Bill of Materials

Items Description	QTY	Unit Cost	Ext. Cost	Vendor
Rpi4 2Gb	1	$55.99	$55.99	Mcmaster Carr
SD Card 64 gb	1	15.63	$15.63	Mcmaster Carr
PLA Fillament	1	$12.00	$12.00	MatterHackers.com
PetG Fillament	1	$13.00	$13.00	MatterHackers.com
USB Bulkhead connections	2	69.67	$139.34	Mouser
20 Gallon Black Polyethylene Tank - 18" L x 12" W x 24" Hg	1	120.67	$120.67	United states Plastic Corp.
18" L x 12" W Black Standard Tamco® Tank Cover (item 14	1	33.29	$33.29	United states Plastic Corp.
3/4" Black Long Thread Water Butt Tap with Gasket (item 1	1	$2.27	$2.27	United states Plastic Corp.
3/4" Black Nut for Water Butt Tap(item 17532)	1	$0.96	$0.96	United states Plastic Corp.
Machining and Powder coating for Enclosure	1	$497.75	$497.75	Emtec Engineering
Panel for electronics mounting	1	$10.02	$10.02	McMaster Carr
USB cable from comm. board to RPi	1	$13.53	$13.53	McMaster Carr
USB to Bulkhead Connectors, Antenna & Spectrometer	2	$14.79	$29.58	Showmecables.com
USB from bulkhead to Spectrometer	1	$9.45	$9.45	Showmecables.com
Plug for external USB connections	2	$62.04	$124.08	Mouser.com
Wifi Antenna	1	$52.99	$52.99	Walmart
RS485 cable	1	40	$40.00	Global FIA
Pumps with Mdrive+ Nema17 Intergrated Controller	2	$2,938.00	$5,876.00	Global FIA
Communication Board, Flo Pro Uni	1	$485.00	$485.00	Global FIA
Selector valve	1	$1,638.00	$1,638.00	Global FIA
TIP 120	2	$5.00	$10.00	Global FIA
1k Ohm Resistor	2	$1.00	$2.00	Global FIA
Header and ProtoBoard	1	$5.00	$5.00	Global FIA
Fittings and Tubing	1	$50.00	$50.00	Global FIA
Misc				
	total		$9,236.55	