Logo, company name

Description automatically generated

Senior Design Documentation

***FLASC***

***(Fast Liquid Analysis and Sanitization Container)***

Prepared by:

**Group #5**

***Self-Sponsored***

**Neeil Gandhi**   **(Photonic Science and Engineering)**

**Ryan Koons**   **(Electrical Engineering)**

**Dean Pickett**   **(Electrical Engineering)**

**Matthew Woodruff (Photonic Science and Engineering)**

Submission Date: 12/04/2021

Dr. Samuel Richie & Dr. Aravinda Kar

EEL 4915L, Fall 2021

Department of Electrical & Computer Engineering

CREOL, The College of Optics and Photonics

University of Central Florida

Table of Contents

[1 Executive Summary 1](#_Toc89545242)

[2 Initial Project Description 2](#_Toc89545243)

[2.1 Project Narrative and Motivation 2](#_Toc89545244)

[2.2 Project Components 2](#_Toc89545245)

[2.2.1 Water Quality Sensor 2](#_Toc89545246)

[2.2.2 Sanitization 3](#_Toc89545247)

[2.2.3 Device Power 3](#_Toc89545248)

[2.2.4 Embedded Hardware 3](#_Toc89545249)

[2.2.5 Software 4](#_Toc89545250)

[3 Requirements 5](#_Toc89545251)

[4 Prototype Model 8](#_Toc89545252)

[5 Block Diagrams and Layouts 9](#_Toc89545253)

[5.1 Water Clarity Sensor 9](#_Toc89545254)

[5.2 Sanitization 11](#_Toc89545255)

[5.3 Device Power 14](#_Toc89545256)

[5.4 Embedded Hardware 15](#_Toc89545257)

[5.5 Software 16](#_Toc89545258)

[6 Estimated Budget and Financing 17](#_Toc89545259)

[7 Initial Project Milestones 18](#_Toc89545260)

[8 Engineering-Marketing Tradeoff 20](#_Toc89545261)

[9 Technology Investigation for Project Development 22](#_Toc89545262)

[9.1 Device Power 22](#_Toc89545263)

[9.1.1 Inductive Charging 22](#_Toc89545264)

[9.1.2 Magnetic Resonance Charging 23](#_Toc89545265)

[9.1.3 Battery 24](#_Toc89545266)

[9.1.4 Supply Voltage Regulators 25](#_Toc89545267)

[9.2 Sanitization 26](#_Toc89545268)

[9.2.1 Sanitization Methods 27](#_Toc89545269)

[9.2.2 Safety and Precautions of Ultraviolet Light 31](#_Toc89545270)

[9.2.3 Different types of UV Sources 31](#_Toc89545271)

[9.2.4 LED 33](#_Toc89545272)

[9.2.5 Technology in the Industry 34](#_Toc89545273)

[9.2.6 Tunable UV spectrum and it’s efficiency 36](#_Toc89545274)

[9.2.7 Selecting the LEDs 37](#_Toc89545275)

[9.2.8 LED Design 39](#_Toc89545276)

[9.2.9 LED Feedback Mechanism 40](#_Toc89545277)

[9.3 Water Clarity Analysis 41](#_Toc89545278)

[9.3.1 Methodology 41](#_Toc89545279)

[9.3.2 Spectral Isolators (Obsolete) 43](#_Toc89545280)

[9.3.3 Optical Detector 44](#_Toc89545281)

[9.3.4 Rotational Mechanism (Obsolete) 45](#_Toc89545282)

[9.4 Safety Switch Detection Device 47](#_Toc89545283)

[9.4.1 Reed Switch 48](#_Toc89545284)

[9.4.2 Hall Effect Switch 48](#_Toc89545285)

[9.4.3 Safety Switch Type Selection 48](#_Toc89545286)

[9.4.4 Reed Switch Selection 48](#_Toc89545287)

[9.4.5 Magnet Selection 50](#_Toc89545288)

[9.5 Embedded Hardware 50](#_Toc89545289)

[9.5.1 Controller Selection 50](#_Toc89545290)

[9.5.2 Controller Selection Summary 53](#_Toc89545291)

[9.5.3 Spectrometer V1 Project Implementation 54](#_Toc89545292)

[9.5.4 Optical Diode PWM Potential Implementation 54](#_Toc89545293)

[9.5.5 Microcontroller Comparisons 55](#_Toc89545294)

[9.5.6 Microcontroller Development 64](#_Toc89545295)

[9.5.7 Bluetooth Technology Comparisons 65](#_Toc89545296)

[9.5.8 Bluetooth Module Investigation 66](#_Toc89545297)

[9.5.9 CYBLE-013025-EVAL EZ-BLE™ Module Arduino Evaluation Board 69](#_Toc89545298)

[9.5.10 EZ-Serial WICED BLE Firmware Platform 70](#_Toc89545299)

[9.5.11 Bluetooth Module Hardware Implementation 72](#_Toc89545300)

[9.5.12 Wired Communication Protocols 72](#_Toc89545301)

[9.6 Microcontroller Software Implementation 74](#_Toc89545302)

[9.6.1 GPIO Pins 74](#_Toc89545303)

[9.6.2 GPIO Interrupt (LLWU) Pins 74](#_Toc89545304)

[9.6.3 Interrupt Service Routines (ISRs) 75](#_Toc89545305)

[9.6.4 Non-GPIO Pin Configuration 75](#_Toc89545306)

[9.6.5 MSP430 Driverlib 75](#_Toc89545307)

[9.6.6 Timers 76](#_Toc89545308)

[9.6.7 Low Power Mode 76](#_Toc89545309)

[9.7 Structural Design/Device Housing 77](#_Toc89545310)

[9.7.1 Original Prusa i3 MK3 (3D Printer) 77](#_Toc89545311)

[9.7.2 FreeCAD 77](#_Toc89545312)

[9.7.3 PrusaSlicer 77](#_Toc89545313)

[9.7.4 Prusament (Filament) 78](#_Toc89545314)

[9.8 Application Software 78](#_Toc89545315)

[9.8.1 iOS App Development 78](#_Toc89545316)

[9.8.2 Android App Development 78](#_Toc89545317)

[9.8.3 Object-Oriented Programming Basics 79](#_Toc89545318)

[9.8.4 Android BLE App Development 80](#_Toc89545319)

[9.8.5 CySmart App 82](#_Toc89545320)

[9.9 Version Control Systems 83](#_Toc89545321)

[9.9.1 GitHub 83](#_Toc89545322)

[9.9.2 Azure DevOps 83](#_Toc89545323)

[9.9.3 AWS Code Commit 83](#_Toc89545324)

[9.9.4 Version Control Summary 83](#_Toc89545325)

[9.10 User Operation 84](#_Toc89545326)

[9.10.1 Ice Cube Usage 84](#_Toc89545327)

[9.10.2 Protection from Impact 84](#_Toc89545328)

[9.10.3 Water Temperature Range 84](#_Toc89545329)

[10 Project Management 86](#_Toc89545330)

[10.1 Software Development 86](#_Toc89545331)

[10.1.1 Shared Codebase 86](#_Toc89545332)

[10.1.2 Issue Reporting/Assignments 86](#_Toc89545333)

[10.1.3 Proper Programming Practices 86](#_Toc89545334)

[10.2 Virtual Breadboard Prototype 86](#_Toc89545335)

[11 Power Distribution Design 88](#_Toc89545336)

[11.1 Voltage Regulators 90](#_Toc89545337)

[11.2 LDO Linear Regulator Addendum 92](#_Toc89545338)

[12 Embedded Hardware Design 94](#_Toc89545339)

[12.1 Utilizing EAGLE Software 94](#_Toc89545340)

[12.1.1 Microcontroller 95](#_Toc89545341)

[12.1.2 Device Sub-system Connections 95](#_Toc89545342)

[12.1.3 Pin Configurations 97](#_Toc89545343)

[12.1.4 Final Pin Configuration 100](#_Toc89545344)

[12.1.5 Spy-Bi-Wire Programming 102](#_Toc89545345)

[12.2 Preliminary Schematic 102](#_Toc89545346)

[12.2.1 Voltage Regulators 103](#_Toc89545347)

[12.2.2 Microcontroller 105](#_Toc89545348)

[12.2.3 Battery Circuit 106](#_Toc89545349)

[12.2.4 Indicator LEDs 107](#_Toc89545350)

[12.2.5 Push Buttons 107](#_Toc89545351)

[12.2.6 Laser Diode and Analog I/Ps 108](#_Toc89545352)

[12.2.7 UVCs and Servomotor (Obsolete) 109](#_Toc89545353)

[12.2.8 Bluetooth Module 110](#_Toc89545354)

[12.2.9 Reed Switch 110](#_Toc89545355)

[12.2.10 Microcontroller Flashing/Programming Header 111](#_Toc89545356)

[12.3 Final Rev. 1 Schematics 112](#_Toc89545357)

[12.3.1 Voltage Regulators 112](#_Toc89545358)

[12.3.2 Microcontroller 113](#_Toc89545359)

[12.3.3 Battery Circuit 114](#_Toc89545360)

[12.3.4 Indicator LEDs 115](#_Toc89545361)

[12.3.5 Push Buttons 115](#_Toc89545362)

[12.3.6 Laser Diode 116](#_Toc89545363)

[12.3.7 Analog Inputs 118](#_Toc89545364)

[12.3.8 UVCs and Motor 118](#_Toc89545365)

[12.3.9 Bluetooth Module 119](#_Toc89545366)

[12.3.10 Reed Switch 120](#_Toc89545367)

[12.3.11 Programming Header 120](#_Toc89545368)

[12.3.12 Comprehensive Rev. 1 Schematic 121](#_Toc89545369)

[12.4 Printed Circuit Board Design 122](#_Toc89545370)

[12.5 PCB and Component Assembly 123](#_Toc89545371)

[13 Water Clarity System Design 126](#_Toc89545372)

[13.1 Spectrometer V2 Project Implementation 126](#_Toc89545373)

[13.2 Spectrometer V3 Project Implementation 126](#_Toc89545374)

[13.3 Water Clarity System Project Implementation 127](#_Toc89545375)

[14 Embedded Firmware Design 128](#_Toc89545376)

[14.1 Design Overview 128](#_Toc89545377)

[14.2 Main.c 128](#_Toc89545378)

[14.3 Reed.c 129](#_Toc89545379)

[14.4 Sanitize.c 130](#_Toc89545380)

[14.5 Analyze.c 131](#_Toc89545381)

[14.6 Export.c 132](#_Toc89545382)

[14.7 BatteryRead.c 132](#_Toc89545383)

[15 Prototyping and Breadboards 134](#_Toc89545384)

[15.1 UVC Sanitization 134](#_Toc89545385)

[15.2 Bluetooth Module 134](#_Toc89545386)

[15.3 Analyzer Prototyping 137](#_Toc89545387)

[15.3.1 Water Quality System v2 (Obsolete) 137](#_Toc89545388)

[15.3.2 Water Clarity Analyzer 139](#_Toc89545389)

[16 Device Operation 140](#_Toc89545390)

[16.1 Sample Preparation 140](#_Toc89545391)

[16.2 General Device Operation 140](#_Toc89545392)

[16.3 Using the App 142](#_Toc89545393)

[16.4 Charging 143](#_Toc89545394)

[16.5 Recommendations for Prolonged Lifetime 144](#_Toc89545395)

[17 Testing 145](#_Toc89545396)

[17.1 Components 145](#_Toc89545397)

[17.2 Systems 148](#_Toc89545398)

[17.3 Programs 150](#_Toc89545399)

[17.3.1 Servomotor Optical Demo Code (Obsolete) 150](#_Toc89545400)

[17.3.2 Stepper Motor Code (Obsolete) 151](#_Toc89545401)

[18 Facilities and Equipment 152](#_Toc89545402)

[18.1 Facilities 152](#_Toc89545403)

[18.2 Equipment 152](#_Toc89545404)

[19 Constraints and Standards 154](#_Toc89545405)

[19.1 Realistic Design Constraints 154](#_Toc89545406)

[19.1.1 Economic and Time Constraints 154](#_Toc89545407)

[19.1.2 Environmental, Social, and Political Constraints 155](#_Toc89545408)

[19.1.3 Ethical, Health, and Safety Constraints 156](#_Toc89545409)

[19.1.4 Manufacturability and Sustainability Constraints 157](#_Toc89545410)

[19.2 Standards 158](#_Toc89545411)

[20 Specifications 160](#_Toc89545412)

[21 Structural Diagram 162](#_Toc89545413)

[22 Biographies 163](#_Toc89545414)

[23 Summary and Conclusions 164](#_Toc89545415)

[24 References 167](#_Toc89545416)

[25 Acknowledgements 173](#_Toc89545417)

[26 Appendix A: Datasheets 174](#_Toc89545418)

[27 Appendix B: Copyrights and Permission 175](#_Toc89545419)

[28 Appendix C: Project Spending 178](#_Toc89545420)

|  |
| --- |
|  |

Table of Figures

[Figure 1: Initial Prototype Model 8](#_Toc89545421)

[Figure 2: Block Diagram Assignment 9](#_Toc89545422)

[Figure 3: Water Quality Sensor Block Diagram (Obsolete) 9](#_Toc89545423)

[Figure 4: Compact Circular Spectrometer V3 10](#_Toc89545424)

[Figure 5: Water Clarity Sensor Block Diagram 10](#_Toc89545425)

[Figure 6: Water Clarity Cutout 11](#_Toc89545426)

[Figure 7: Sanitization Block Diagram (Obsolete) 12](#_Toc89545427)

[Figure 8: Implemented Sanitization Block Diagram 12](#_Toc89545428)

[Figure 9: Original Optical Sanitizer Design (Obsolete) 13](#_Toc89545429)

[Figure 10: Implemented optical Sanitizer Design with Feedback System 13](#_Toc89545430)

[Figure 11: Device Power Block Diagram 14](#_Toc89545431)

[Figure 12: Embedded Hardware Block Diagram 15](#_Toc89545432)

[Figure 13: Software Block Diagram 16](#_Toc89545433)

[Figure 14: House of Quality 21](#_Toc89545434)

[Figure 15: Electromagnetic Spectrum Figure taken from [17] 29](#_Toc89545435)

[Figure 16. Showing some of the examples of water disinfectant in the industry. Figure taken from [22]. 35](#_Toc89545436)

[Figure 17: SpecialRED Sanitizer. Figure taken from [24]. 35](#_Toc89545437)

[Figure 18. Single LED. 36](#_Toc89545438)

[Figure 19. Spectral sensitivity of adenovirus 2 and its DNA damage. Figure from [28]. 37](#_Toc89545439)

[Figure 20: Power Distribution Diagram 88](#_Toc89545440)

[Figure 21: Preliminary 3V Switching Regulator Circuit 103](#_Toc89545441)

[Figure 22: Preliminary +3V LDO Regulator Schematic 104](#_Toc89545442)

[Figure 23: Preliminary +6V Switching Regulator Schematic 104](#_Toc89545443)

[Figure 24: Preliminary Microcontroller Schematic 105](#_Toc89545444)

[Figure 25: Preliminary Power Coupling Schematic 106](#_Toc89545445)

[Figure 26: Preliminary Battery Circuit Schematic 106](#_Toc89545446)

[Figure 27: Preliminary Indicator LEDs Schematic 107](#_Toc89545447)

[Figure 28: Preliminary Push Buttons Schematic 108](#_Toc89545448)

[Figure 29: Preliminary Laser Diode and Analog I/Ps Schematic 109](#_Toc89545449)

[Figure 30: Preliminary UVCs and Servomotor Schematic 109](#_Toc89545450)

[Figure 31: Preliminary Bluetooth Module Schematic 110](#_Toc89545451)

[Figure 32: Preliminary Reed Switch Schematic 111](#_Toc89545452)

[Figure 33: Preliminary Programming Header Schematic 111](#_Toc89545453)

[Figure 34: +3V LDO Regulator Schematic 112](#_Toc89545454)

[Figure 35: +6V Switching Regulator Schematic 113](#_Toc89545455)

[Figure 36: Microcontroller Schematic 114](#_Toc89545456)

[Figure 37: Battery Circuit 115](#_Toc89545457)

[Figure 38: Indicator LEDs Schematic 115](#_Toc89545458)

[Figure 39: Push Buttons Schematic 116](#_Toc89545459)

[Figure 40: Laser Diode Schematic 117](#_Toc89545460)

[Figure 41:Analog Inputs Schematic 118](#_Toc89545461)

[Figure 42: UVCs and Motor Schematic 119](#_Toc89545462)

[Figure 43: Bluetooth Module Schematic 119](#_Toc89545463)

[Figure 44: Reed Switch Schematic 120](#_Toc89545464)

[Figure 45: Programming Header Schematic 120](#_Toc89545465)

[Figure 46: Comprehensive Rev. 1 Schematic 121](#_Toc89545466)

[Figure 47: Final Printed Circuit Board Layout--TOP 122](#_Toc89545467)

[Figure 48: Final Printed Circuit Board—BOTTOM 123](#_Toc89545468)

[Figure 49: FLASC Printed Circuit Board 125](#_Toc89545469)

[Figure 50: Main.c Flowchart 129](#_Toc89545470)

[Figure 51: Reed.c Flowchart 130](#_Toc89545471)

[Figure 52: Sanitize.c Flowchart 131](#_Toc89545472)

[Figure 53: Analyzer.c Flowchart 132](#_Toc89545473)

[Figure 54: Sanitizer Optical Setup 134](#_Toc89545474)

[Figure 55: EZ-BLE Eval Board Connections 135](#_Toc89545475)

[Figure 56: RealTerm Bluetooth Transmission Example 137](#_Toc89545476)

[Figure 57: Water Quality Sensor Board Prototype 138](#_Toc89545477)

[Figure 58: Optical Level (Obsolete) 138](#_Toc89545478)

[Figure 59: Water clarity analyzer prototype layout. 139](#_Toc89545479)

[Figure 60: FLASC power switch (left) and LED indicator (right) locations. 141](https://knightsucfedu39751-my.sharepoint.com/personal/rkoons17_knights_ucf_edu/Documents/SD%20Project/SD2%20Group%205%20Final%20Document.docx#_Toc89545480)

[Figure 61: Sanitizer (left) and analyzer (right) buttons. 142](https://knightsucfedu39751-my.sharepoint.com/personal/rkoons17_knights_ucf_edu/Documents/SD%20Project/SD2%20Group%205%20Final%20Document.docx#_Toc89545481)

[Figure 62: Wireless Charging Orientation 143](#_Toc89545482)

[Figure 63: UV-C LED measured spectrum. 148](#_Toc89545483)

[Figure 64: Structural Diagram 162](#_Toc89545484)

|  |
| --- |
|  |

Table of Tables

[Table 1: Requirements—Performance 5](#_Toc89545485)

[Table 2: Requirements—Physical Characteristics/Dimensions 5](#_Toc89545486)

[Table 3: Requirements—Sanitization 5](#_Toc89545487)

[Table 4: Requirements—Water Quality Sensor 6](#_Toc89545488)

[Table 5: Requirements—Device Power 6](#_Toc89545489)

[Table 6: Requirement—Embedded Hardware 6](#_Toc89545490)

[Table 7: Requirements—Wireless Communication 7](#_Toc89545491)

[Table 8: Requirements—Software/App 7](#_Toc89545492)

[Table 9: Estimated Budget 17](#_Toc89545493)

[Table 10: Initial Project Milestones Legend 18](#_Toc89545494)

[Table 11: Senior Design 1 Milestones 18](#_Toc89545495)

[Table 12: Senior Design 2 Milestones 19](#_Toc89545496)

[Table 13: Sanitization Methods Comparison 31](#_Toc89545497)

[Table 14: Comparison between UV LED and UV Lamp. Referenced from 21. 33](#_Toc89545498)

[Table 15: Summary of Methodologies 43](#_Toc89545499)

[Table 16: Summary of Isolators 44](#_Toc89545500)

[Table 17: Summary of Sensors 45](#_Toc89545501)

[Table 18: Summary of Rotational Mechanisms 47](#_Toc89545502)

[Table 19 Summary of Reed Switch Options 49](#_Toc89545503)

[Table 20: Summary of Magnet Options 50](#_Toc89545504)

[Table 21: Controller Selection Summary 54](#_Toc89545505)

[Table 22: Microcontroller Feature Overview 55](#_Toc89545506)

[Table 23: Proposed G2553 20-Pin Configuration 58](#_Toc89545507)

[Table 24: Microcontroller Comparisons 1 60](#_Toc89545508)

[Table 25: Microcontroller Comparisons 2 63](#_Toc89545509)

[Table 26: Bluetooth Module Comparisons 69](#_Toc89545510)

[Table 27: Default UART settings for Cypress BLE module 70](#_Toc89545511)

[Table 28: Summary of Voltage and Current Requirements 89](#_Toc89545512)

[Table 29: FLASC Component Energy Usage Estimate 90](#_Toc89545513)

[Table 30: Comparison of 3.0V Buck-Boost Converter Designs 91](#_Toc89545514)

[Table 31: Comparison of 6.0V Boost Converter Designs 92](#_Toc89545515)

[Table 32: Water Quality Sensor Connections 96](#_Toc89545516)

[Table 33: Sanitization Connections 96](#_Toc89545517)

[Table 34: Bluetooth Connections 96](#_Toc89545518)

[Table 35: Microcontroller Flashing Connections 96](#_Toc89545519)

[Table 36:LED Indicators Connections 97](#_Toc89545520)

[Table 37: Pin Configuration 1 98](#_Toc89545521)

[Table 38: Development Board External Headers 99](#_Toc89545522)

[Table 39: Pin Configuration 2 100](#_Toc89545523)

[Table 40: Final Microcontroller Pin Configuration 101](#_Toc89545524)

[Table 41: Indicator LED Electrical Characteristics 107](#_Toc89545525)

[Table 42: Laser Diode Curve Trace Table 116](#_Toc89545526)

[Table 43: Laser Diode Resistor Calculations 117](#_Toc89545527)

[Table 44: Bill of Materials 124](#_Toc89545528)

[Table 45: CYBLE-013025 Pin Connections 135](#_Toc89545529)

[Table 46: Analyzer Component Tests 146](#_Toc89545530)

[Table 47: Laser Diode Electrical Characteristics 146](#_Toc89545531)

[Table 48: Sanitizer Component Tests 147](#_Toc89545532)

[Table 49: Project System Tests 150](#_Toc89545533)

[Table 50: Facilities 152](#_Toc89545534)

[Table 51: Equipment 153](#_Toc89545535)

[Table 52: Environmental Constraints 156](#_Toc89545536)

[Table 53: Manufacturing and Sustainability Constraints 158](#_Toc89545537)

[Table 54: Standards 159](#_Toc89545538)

[Table 55: Analysis/Spectrometer Specifications 160](#_Toc89545539)

[Table 56: Sanitizer Specifications 161](#_Toc89545540)

[Table 57: Power Supply Specifications 161](#_Toc89545541)

[Table 58: Project Spending 178](#_Toc89545542)

# Executive Summary

According to the CDC, between 30 and 70 percent of global explorers suffer from traveler’s sickness. Of those individuals, more than 80 percent fall ill due to bacterial pathogens [1]. One of the most common ways that travelers get exposed to bacteria is through the consumption of biological contaminants in water from underdeveloped sanitation systems. A single sip of improperly treated hydration can expose people to various gruesome diseases including typhoid, cholera, salmonellosis, and a myriad of unpleasant maladies [2]. With sterility in the vogue, virus and disease prevention has prime real estate in the minds of much of the population. And, due to a continuous need for water, pure drinking water is integral to personal wellbeing. This project delivers a long-term, portable, and affordable solution for consumers seeking assurance about their water’s quality. A water bottle capable of providing information about the drinkability of its contents would be very useful for an array of circumstances from international travel to preparedness for natural disaster, but it also applies to those merely going about their daily routine.

It is quite common practice in the industry today to utilize optical light to sanitize drinking water and make it more potable- indeed, optical sanitization is a quick and convenient method for anyone to sanitize their drinking water. In addition to this, our FLASC (Fast Liquid Analysis and Sanitization Container) product incorporates water clarity analysis as an indicator of water quality. Our prototype is a smart water bottle that can sanitize its contents using UV irradiation and also analyze its sample for water clarity, both the visible and invisible (infrared regime). The sanitization process commences whenever the user initiates the procedure via a button on the bottle, and after sanitization is complete the user is provided a clarity analysis which can factor in to a judgement call as to whether or not the liquid is safe. This information is displayed to the user through LED indicators, and the water clarity is also sent to an app alongside the battery percentage.

It is true that one intent behind this product was to ensure peace-of-mind for international travelers concerned by water-borne pathogens and parasites. However, the FLASC should also appeal to the tech savvy consumer that likes to have “smart” devices with added functionality for ease in daily life. Some of this added functionality includes wireless charging to simplify daily device operation, Bluetooth connectivity, and app access to battery life information and water clarity reports.

# Initial Project Description

## Project Narrative and Motivation

The project's final device contains several systems from a technology standpoint. It has wireless charging built into the cap to allow for convenient recharges of the device as well as improved water resistance. The optics system consists of UVC diodes for sanitizing biological contaminants and a feedback system to verify that sanitization is occurring. The bottle also has a laser and phototransistor for determining the content’s clarity as an indicator of water quality. The bottle has multiple LEDs that serve as status and result indications, a Bluetooth module for communications with a paired device, a reed switch to ensure the sanitizer and analyzer only runs in appropriate conditions, and an embedded microcontroller. All of the electronics are contained inside a bottle-top mounted device comparable in size to a soda can and which is compatible with other metal water bottles with similar cap designs. The Bluetooth functionality pairs with the CySmart app to display the battery percentage and a Boolean description of water clarity. The intent behind our design was to minimize costs and complexity for the user interface by eliminating a built-in screen and instead repurposing one the user likely already has while still maintaining functionality without said screen.

Most smart bottles in the market today implement less useful features such as glowing to remind you when to drink, a built-in Bluetooth speaker, and water intake tracking. Some bottles are also UV sanitizing but tend to run on a schedule and do not allow the user to simply press a button to initiate sanitization. Other senior design smart water bottle projects also had water sanitization and smart features such as water intake and a Bluetooth app- however, these projects did not incorporate wireless charging or water analysis. Previous projects also lack modularity, a feature of significant importance as the proposed device can be switched with multiple compatible bottle units, allowing a group of users to purchase only a single FLASC unit alongside many units of the bottle, reducing their investment without impacting utility.

## Project Components

### Water Quality Sensor

This component was, in the initial design, intended to perform spectral analysis of the contained fluids. The most nearly realized version would have consisted of a spectrometer that observed scattering effects of monochromatic light to deduce the presence of contaminants. Mechanically, the spectrometer was to use a photodiode on a linear stepper motor to step through the spectrum, which would be dispersed by a prism. However, a combination of equipment breakage and repeated shipping delays led to the abandonment of this design. Another option would use a photodiode and rotational mechanism to step through a range of wavelengths, recording at intervals.

The water quality analyzer was quickly re-imagined as a water clarity sensor, which uses the previously acquired green laser diode (with infrared leakage) to bounce a beam off the bottom of the bottle, through a viewing port, and onto a phototransistor, allowing for a measure of the water clarity. This implementation has advantages over the initial spectrometer design as it is steady-state, economical, and less complex yet still capable of detecting certain contaminants which are difficult to discern with the naked eye (principally 4 ppm concentrations of fluoride, the maximum recommended concentration as listed by the EPA [3]).

### Sanitization

This component sterilizes samples of water in the bottle to ensure user safety. The components for this are placed in the bottle cap we designed. We decided to go with 2 single UVC LEDs with different wavelengths as it is a good compromise of cost, time and power efficiencies while remaining compact and easy to implement. This implementation produces a broad spectrum of light, and a feedback system ensures that there is enough intensity to sanitize the water completely. According to NIST, a tunable laser would be more efficient at killing biological contaminants in water (and this was the original design for the sanitizer)- however, this would significantly increase the cost and fragility of the FLASC [4]. Our device's UVC light kills biological contaminants present makes it safer to drink the bottle's contents. This helps prevent infections from water-borne diseases such as typhoid and cholera.

### Device Power

The device is powered by a rechargeable battery. Of the initial options which were investigated (namely, Nickel-Metal Hydride and Lithium-Ion battery technologies), a decision was made to utilize Lithium-Polymer. The charged battery supplies power to the other components by relaying power through 6V and 3V DC-DC voltage regulators to ensure that the FLASC's sensitive electrical components receive their intended voltages within a tight tolerance and with protection from shorts and overheating.

Battery charging is a simple task with the FLASC's wireless transmission of AC voltage supplied by a standard US 120 Volt wall outlet. FLASC can recharge via a wireless power transmitter, which connects inductively to the device's receiver coil through the plastic cap, creating an effective and convenient charging system with the ability to fully recharge a drained unit overnight without the struggle (and water hazard) of USB ports. As many people already have at least one wireless charging dock due to the rise in popularity of wirelessly charged phones and other accessories, the FLASC was designed to charge off of any Qi-compliant wireless charging pad.

### Embedded Hardware

The device hardware mainly involved sensors, LEDs, a microcontroller, and a Bluetooth module. This includes a water clarity system, photoresistor, sanitizing LEDs, and a reed switch for verifying that the FLASC unit is safely in place during operation. The reed switch was especially important as UVC light is harmful to human beings and therefore needed to be contained inside the bottle. All of these sensors are controlled by the microcontroller. It was preferable to use a TI microcontroller due to the familiarity of embedded programming software. The microcontroller also had to have enough I/O pins to support the sensors (reed switch included), LEDs, the Bluetooth module, and (previously) drive either a servomotor or stepper motor. A microcontroller with a low power mode was strongly desired to improve the battery life of the product, as well as low level wake up functionality with device peripherals. It also had to be compatible with common communication protocols such as UART and I2C (perhaps also SPI). The microcontroller (even with high-drive current pins) was not able to directly drive enough current to the laser (or the obsolete motors), so a MOSFET switching circuit was implemented for enabling/disabling the UVC laser diode. A red/green/yellow LED (in one package) was used to provide rudimentary water clarity information to the user post-sanitization and to display the current device status mode. It was proposed to include an Integrated Circuit to provide a dot/bar voltage display that shows the charge status of the battery in an array of 10 LEDs [5]. Due to time constraints and project overhead, this was dropped in favor of transmitting a more exact battery estimation to an app. FLASC has a button to start sanitization and another for checking the water clarity. The analysis button can be held down for approximately two seconds to export the latest water clarity verdict and battery life estimation from FLASC. Other features were added later on in project development and are described in greater detail throughout this document. From a design standpoint: EAGLE PCB design software was utilized due to ongoing familiarity, and the boards were ordered and assembled by PCBWay for the same reasons. A turn-key assembly was utilized due to the current industry-wide chip shortage and to ensure reliability in the final product.

### Software

The software was mainly comprised of embedded firmware for the board itself, however a considerable amount of time was also spent on rudimentary app development. Since this project was mainly comprised of Electrical Engineering and Optical majors, eventually a pre-made app was utilized to display wirelessly transmitted data. Code Composer studio was utilized for creating the firmware for the microcontroller. The firmware needed to be able to check that the cap was on before activating sanitization, enable/disable sanitization, check the clarity of the water, display rudimentary water quality information and device modes via LEDs, wirelessly communicate via Bluetooth, and operate in low power modes with interrupt handling for the control logic. It required a significant amount of time to learn about the various sensors and how to interface with them (via the microcontroller). The Bluetooth module required implementation of the UART communication protocol. A lot of development overhead also went into learning the EZ-Serial interface and how to transmit data from a FLASC unit to the user’s smartphone. It also was necessary to create code that keeps track of water clarity and sanitization data, as well as track control logic and safety features. When the device is not readily being used, the microcontroller needed to enter low power mode to conserve energy.

Originally, the custom app was intended to receive data via Bluetooth from the device to report more in-depth water quality data to the user. Using a photodiode, laser diode, and motor, the data would have primarily consist of a vector of the sample’s spectrum. The vector was intended to be produced by moving the motor and sweeping through the spectrum. Preliminary code was written to accomplish this task which would more effectively detect contaminants in the water. However, due to unforeseen circumstances with product availability and shortages, the analyzer was converted into a water clarity system that functioned similarly but without a motor. In this final implementation, a laser diode was turned on, and a photodiode threshold was set to be able to distinguish clear and unclear water.

# Requirements

The requirements sections details project requirements chosen based on expected customer demands, regulatory standards, and technological limitations. Table 1 covers the sanitizer and analyzer's black-box performance requirements and Table 2 outlines a set of physical dimensions which should prove desirable to consumers. Table 3 covers basic features of the sanitizer.

Performance

|  |  |
| --- | --- |
| Maximum shutoff delay following reed switch disconnect | 1 second |
| Time for one sanitization cycle | 3 minutes |
| Sanitization effectivity | Kills 99.9% of contained micro-organisms |
| Sensor detects the presence of fluoride at the EPA's specified concentration [3] | 4 ppm |

Table 1: Requirements—Performance

Physical Characteristics/Dimensions

|  |  |
| --- | --- |
| Water Bottle Lid | Round 3 1/4” diameter x 5” tall  Plastic top to allow for inductive charging |
| Water Bottle Body | Round 3” diameter x 8” tall |
| Waterproof | All circuitries should be contained within a housing which could feasibly be made IP44 certified waterproof |

Table 2: Requirements—Physical Characteristics/Dimensions

Sanitization

|  |  |
| --- | --- |
| UVC Emitter | Multiple LED sources to get a broad spectrum. LED with peak wavelength of 265 nm and 275 nm. |
| Photoresistor | For feedback system, to check if the LEDs are producing enough intensity to sanitize the water or not. |

Table 3: Requirements—Sanitization

Continuing on, Table 4 outlines basic aspects of the analyzer, Table 5 provides general requirements for device power, and Table 6 expounds on embedded hardware requirements.

Water Quality Sensor

|  |  |
| --- | --- |
| Overall Sensitivity | The analyzer must pass distilled water and fail straight coffee and any contaminants listed in Table 1 |
| Overall Size | The analyzer must have less than 2" of the device's overall height dedicated solely to it |

Table 4: Requirements—Water Quality Sensor

Device Power

|  |  |
| --- | --- |
| Battery Lifespan | At least 10 cycles of the sanitization and sensor modules. |
| Inductive Charging Speed | Less than 8 hours from drained to charged. |
| Charging Pad Connection | A standard (US) 120 VAC wall outlet and consist of an inductive transmitter coil, an on/off switch, protective housing, and a permanent magnet (for coil alignment) |

Table 5: Requirements—Device Power

Embedded Hardware

|  |  |
| --- | --- |
| Controller | About 10-20 GPIO pins for device operation. Low power mode functionality with current in the tens of microamps. |
| Wireless Communication | BLE: reduced power consumption. OTA data rate: 1 Mbps. with 6ms latency |
| Wired Communication Protocols | UART: Full-duplex serial system. I2C: Device addressing-serial bus |
| Water Quality Sensor | ADC and/or comparator used to detect phototransistor output to the resolution of microvolts |
| Bluetooth Pairing | Device automatically advertises for Bluetooth Low Energy bonding. |

Table 6: Requirement—Embedded Hardware

The final two tables of this section (Table 7 and Table 8) cover the basic requirements of the wireless communication feature and the app features respectively.

Wireless Communication

|  |  |
| --- | --- |
| Connection Distance | Device should be able to pair with a smartphone from a distance of 10 or more feet |
| Pairing Time | The device should take no longer than 30 seconds to pair/connect with a smartphone |

Table 7: Requirements—Wireless Communication

Software/App

|  |  |
| --- | --- |
| Application | Utilizes at least the Android platform, is Bluetooth pairing and compatible  Displays a Boolean indicator of water clarity |
| Interrupt Handling | Reed switch and sanitization/analysis. Events occur within 1 second of a button press. |
| Communication Protocols | I2C, UART, and Bluetooth (most likely 4.0/BLE) |
| Microcontroller low power mode | Enable low power mode to allow for a sleep current of 10 microamps or less |
| Water Clarity | Collects an ADC reading from the analyzer and produces a verdict on water clarity |
| Sanitization Feedback | Verifies that both LEDs are working within 15 seconds of a run start. |

Table 8: Requirements—Software/App

# Prototype Model

The following section will roughly outline the organization of the devices’ subsystems. This was a very early vision of the physical layout but has seen few changes due to its generality and effectiveness as a plan. The very top of the lid (see leftmost side of Figure 1 below) contains the power electronics for the FLASC. This involves the wireless receiver, battery charging circuit, and the battery itself. The next layer down contains the PCB. This layer has the microcontroller, status indicating LEDs (facing outward), and most of the embedded hardware. The next layer holds the water clarity analyzer sensor and serves as a mounting point for the two lower sections while also providing an airgap below the PCB and ports for wiring. The lowest layer consists of the laser, LEDs, photoresistor, and a viewing port for the water clarity sensor.

The device housing was, as planned, 3D printed with customized mounting points for the various components. Originally, the intention was to utilize the bottle's stainless-steel cap as a base- however, this proved excessively difficult, so a 3D printed surface-interface region was also designed, and on a production model this could be produced using a more appropriate material. This allowed the entire assembly to be produced without the need for any drilling or cutting into the printed layers, as holes were added to the CAD files prior to printing. One shortcoming of the final prototype is that we were not able to incorporate transparent covers for the LEDs and buttons for the switches, as time was not permitting. The exact implementation (housing, components, placement, and etc.) is discussed in later sections.

Initial Prototype Model

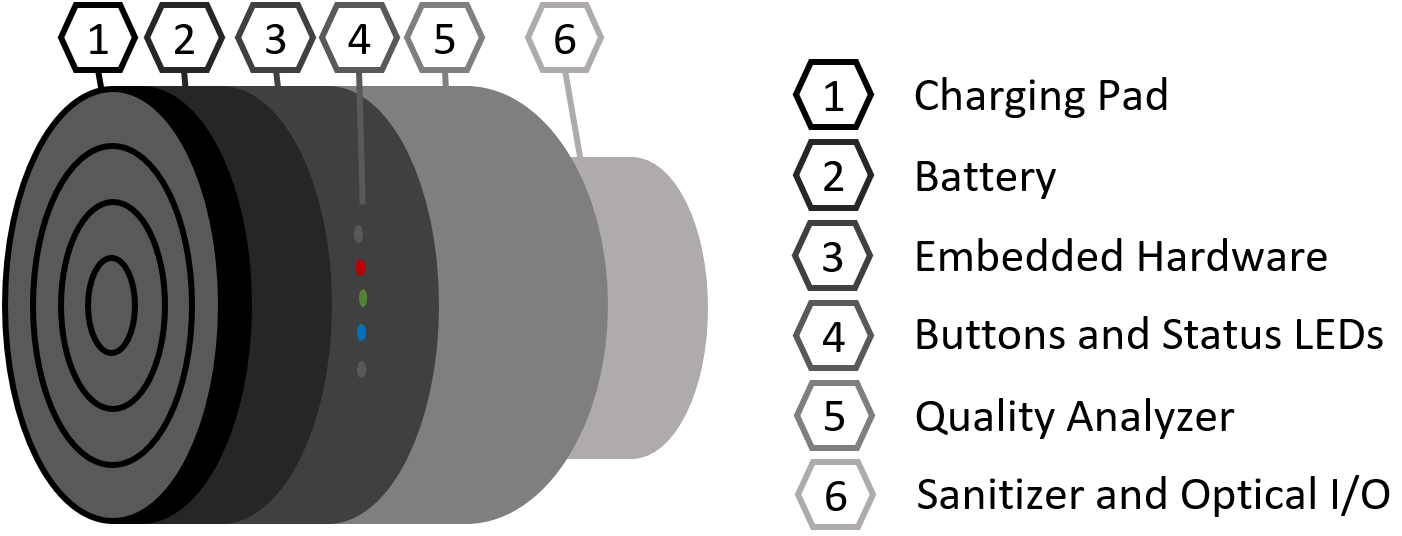


Figure 1: Initial Prototype Model

# Block Diagrams and Layouts

The following section outlines the various sub-systems of the device and their concepts in the project. Each subsystem has individuals primarily responsible for a block diagram and outlines how other team members contributed. The Responsibility Legend seen in Figure 2 below was followed (unless otherwise stated). It was used to show which individual was responsible for a particular block in any of the diagrams. This section also contains preliminary designs for the optical project requirements.

Responsibility Legend

|  |  |
| --- | --- |
| Dean Pickett | Matthew Woodruff |
| Ryan Koons | Neeil Gandhi |

Figure 2: Block Diagram Assignment

## Water Clarity Sensor

Figure 3 details the original process involved in the (obsolete) water quality system.

Figure 4 shows a basic layout of the original system.

Water Quality Sensor Block Diagram (Obsolete)

Figure 3: Water Quality Sensor Block Diagram (Obsolete)

Compact Circular Spectrometer V3 (Obsolete)

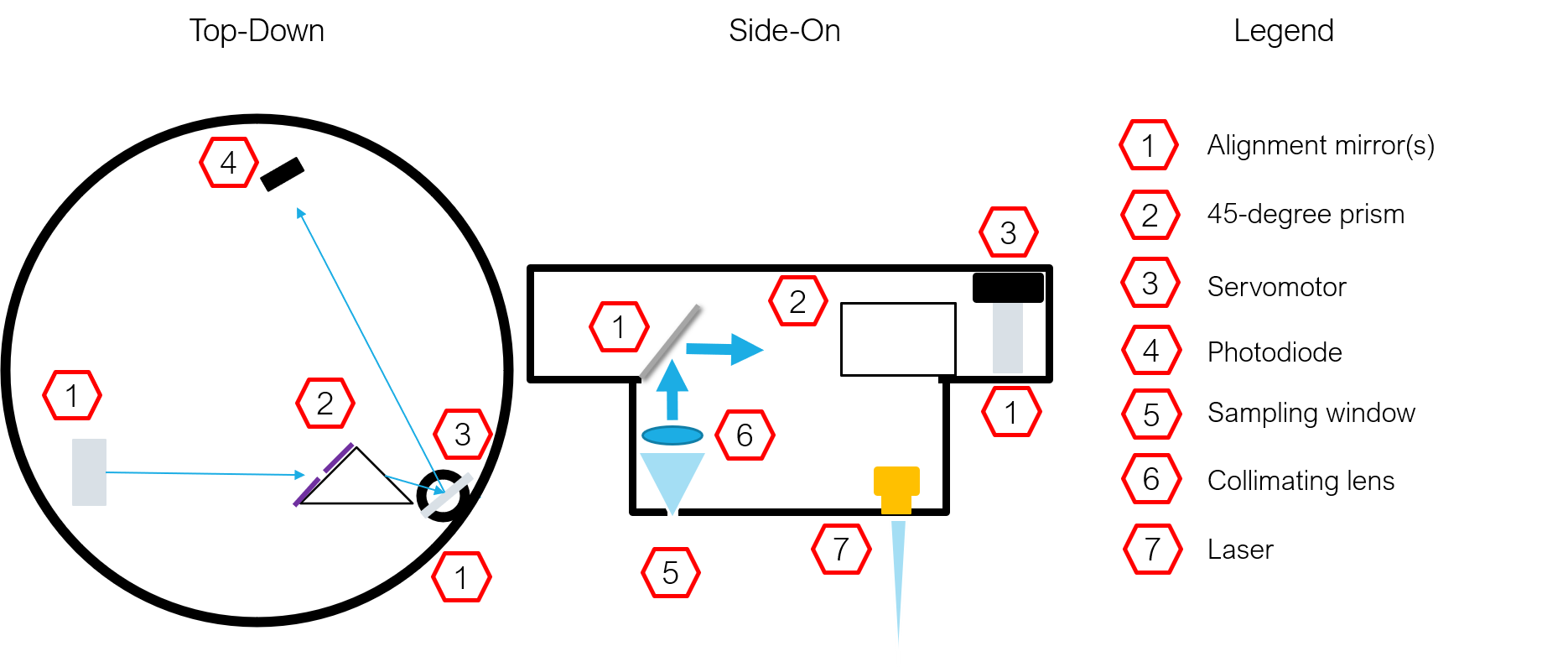


Figure 4: Compact Circular Spectrometer V3

Figure 5 shows the water clarity sensor block diagram and demonstrates the process involved in the clarity sensor.

Water Clarity Sensor Block Diagram

Figure 5: Water Clarity Sensor Block Diagram

Figure 6 shows a cutout of both housing layers (microcontroller layer on top and surface-interface (SI) below) spanned by the analyzer. The design is much simpler- the tube in the SI holds the laser module, the hole in the center of the SI is the viewing window, and the phototransistor rests in the hole above it in the microcontroller layer.

Water Quality Analyzer Cutout

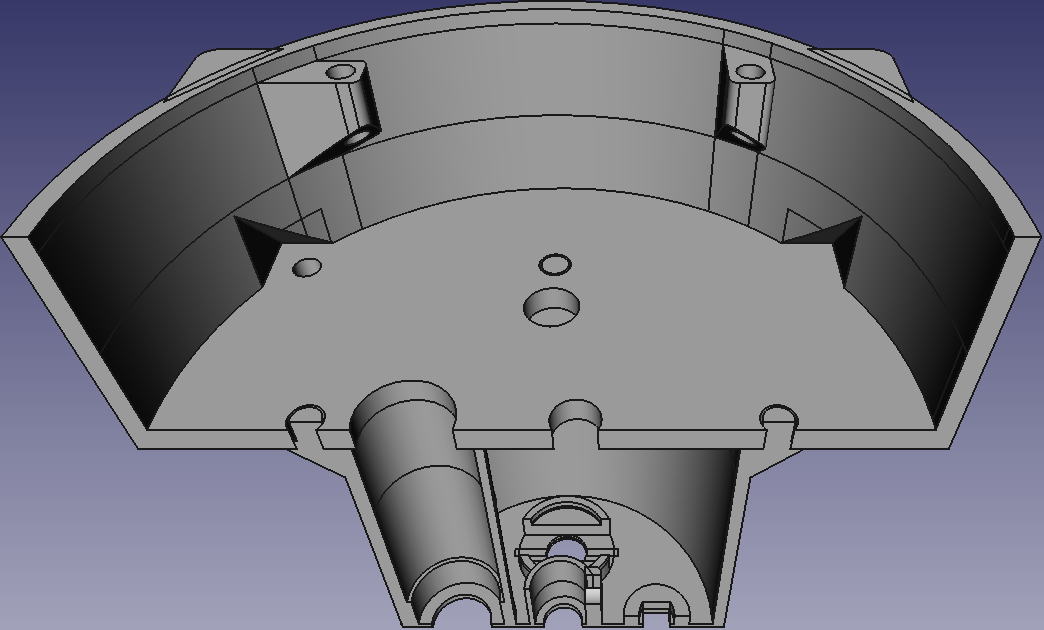


Figure 6: Water Clarity Cutout

## Sanitization

As can be seen below in Figure 9, Neeil is primarily responsible for the sanitizer. See Figure 9 for a physical layout of a device which is obsolete. This was our original design but after testing the components we had to change the design to Figure 10 with adding a feedback loop for the sanitizer. Figure 7 shows the obsolete version of the block diagram which was changed due to optical difficulties to Figure 8, which is implemented in our design.

Sanitization Block Diagram (Obsolete)

Figure : Sanitization Block Diagram (Obsolete)

Implemented Sanitization Block Diagram

Figure 8: Implemented Sanitization Block Diagram

Original Optical Sanitizer Design (Obsolete)

Diagram

Description automatically generated

Figure 9: Original Optical Sanitizer Design (Obsolete)

Implemented Sanitization Design

Diagram

Description automatically generated

Figure 10: Implemented optical Sanitizer Design with Feedback System

## Device Power

Dean was responsible for most of this section (see Figure 11 below), with some assistance from Ryan.

Device Power Block Diagram

Figure : Device Power Block Diagram

## Embedded Hardware

Ryan was mostly responsible for this section (see Figure 12 below) with some assistance from Dean.

Embedded Hardware Block Diagram

Figure : Embedded Hardware Block Diagram

## Software

Ryan was mostly responsible for this section (see Figure 13 below) with some assistance from Dean and Matthew.

Software/App Block Diagram

Figure : Software Block Diagram

# Estimated Budget and Financing

Table 9 below outlines the estimated budget for prototype/development, as well as the typical cost per unit. Production cost (per unit) is an important parameter for this product’s marketability and was therefore factored into all estimated financing.

Estimated Budget

|  |  |  |
| --- | --- | --- |
| Component | Prototype | Production |
| Stainless Steel Double Walled (Generic) Bottle and Lid | $15.00 | $15.00 |
| Quantity of Five: PCB, solder mask, and Assembly (PCBWay). Two separate board revisions ($100.00 per revision) | $200.00 | $20.00 |
| R/G/Y LED | $2.00 | $2.00 |
| Buttons (Sanitization, Water Quality, BT Pairing) | $2.00 | $2.00 |
| Blue LED | $0.50 | $0.50 |
| Reed Switch | $2.00 | $2.00 |
| Wireless module | $10.00 | $10.00 |
| Bluetooth evaluation board | $54.00 | $0.00 |
| UV Sanitizing Diode and photoresistor | $30.00 | $7.00 |
| MOSFETs | $3.00 | $3.00 |
| Water Clarity System (laser, phototransistor) | $25.00 | $25.00 |
| Li-ion/Li-poly charger | $6.00 | $6.00 |
| Qi Compliant Charging Receiver | $15.00 | $15.00 |
| Rechargeable Battery | $20.00 | $20.00 |
| Voltage Regulator Components ($0.60 x3) | $1.80 | $1.80 |
| Microcontroller (10-20 I/O) | $7.00 | $7.00 |
| SMT Passive Components | $5.00 | $5.00 |
| **Approximate Total:** | **$398.30** | **$141.30** |

Table 9: Estimated Budget

# Initial Project Milestones

This section outlines the project milestones and dates from the entirety of senior design. The first table (see Table 10 below) provides a color key for deadline strictness while Table 11 provides an overview of the first semester of senior design and Table 12 provides an overview of the second semester of senior design.

Legend

|  |
| --- |
| SD Hard Deadlines |
| Tentative deadlines |

Table 10: Initial Project Milestones Legend

Senior Design 1 Milestones

|  |  |
| --- | --- |
| **Date** | **Details** |
| 5/21 | Project Conceptual Brainstorming: Optics and Electrical Engineering |
| 5/27 | Project Idea Discussion |
| 5/28 | Project Selection: Decided on Smart Water Bottle |
| 6/9 | Divide & Conquer v1.0: Complete the D&C |
| 6/11 | Divide & Conquer v1.0 Due: Submit the completed D&C |
| 6/15 | D&C V1.0 Meeting with Dr. Richie |
| 6/23 | Divide & Conquer v2.0 Complete: Approximately 25 pages |
| 6/25 | Divide & Conquer v2.0 Due: Submit the final D&C |
| 6/28 | SD1 Paper: 30-page draft completed |
| 7/2 | SD1 Paper: 45-page draft completed |
| 7/5 | SD1 Paper: 60-page draft completed |
| 7/9 | SD1 v1 60-page paper Due |
| 7/16 | SD1 Paper: 80-page draft completed |
| 7/23 | SD1 v2 100-page paper Due |
| 7/30 | Finishing touches |
| 8/3 | Final document due |
| 8/6 | Order prototype boards (PCBWay) |

Table 11: Senior Design 1 Milestones

Senior Design 2 Milestones

|  |  |
| --- | --- |
| 8/23 | Work on critical design review |
| 9/27 | Build and test optical components |
| 9/30 | Critical Design Review is due |
| 10/15 | Project Summary due |
| 10/25 | Work on firmware, and Bluetooth implementation |
| 11/1 | Midterm Demo due |
| 11/12 | Work on conference paper and showcase video, submit committee form. |
| 11/24 | Have everything ready and work on testing, final paper and presentation. |
| 11/26 | Finish and submit showcase video |
| 11/29 | 8-Page Conference Paper, Final PowerPoint Presentation, Final Demo due by 5:00 pm. Showcase (due by 12:00pm). |
| 11/30 | Committee Live Demo and Q&A 11:00am |
| 12/7 | Final Documentation and Website due |

Table 12: Senior Design 2 Milestones

# Engineering-Marketing Tradeoff

All engineering projects require careful consideration and balance between engineering requirements and marketing requirements. With a focus too narrowed on the engineering requirements, the product may become too expensive or difficult to market. Likewise, a product heavily focused on marketability instead of engineering, may not function correctly or function altogether poorly. Thus, it is important during the early stages of this project’s inception to consider all of the various types or requirements and their interaction with one another. The right balance between device requirements will have to be determined such that both types of requirements are sufficiently satisfied. The following diagram below outlines the relationship between engineering and marketing requirements.

Each column of the House of Quality on the next page represents an engineering requirement for the project. Each row represents a marketing requirement for the project. The “Target Specifications” at the bottom of Figure 14 is the value or performance rating related to each engineering requirement. A ‘+’ in the gray section indicates a requirement that should try to be maximized. Whereas a ‘-‘ in the gray section means that a requirement should try to be minimized. The up and down arrows represent the positive or negative correlation (respectively) between each type of requirement. Any comparison that has multiple arrows indicates a stronger correlation between the requirements.

The roof of the house of quality illustrates the relationship between the engineering requirements. This is accomplished by providing a cell for each possible pair of columns and placing an indicator to show whether there is a direct, inverse, or no relation between the factors an upwards-oriented arrow indicates that the two columns share a direct relationship--increasing one column will increase the other. To determine the relationship between two requirements, follow each cell up and diagonally left until they intersect at one point. A simple example of this would be that battery capacity and cost are directly related- a large battery will almost universally cost more than a smaller one (this does not imply, however, that increasing the cost will increase the battery capacity). A downwards-oriented arrow indicates that the two columns share an inverse relation- increasing one column will decrease the other. An example of this would be cost and size as decreasing the overall size of a device, while potentially reducing the total amount of materials needed, tends to require more demanding designing and compact components. An empty cell indicates that the two columns have no relationship with each other.

House of Quality

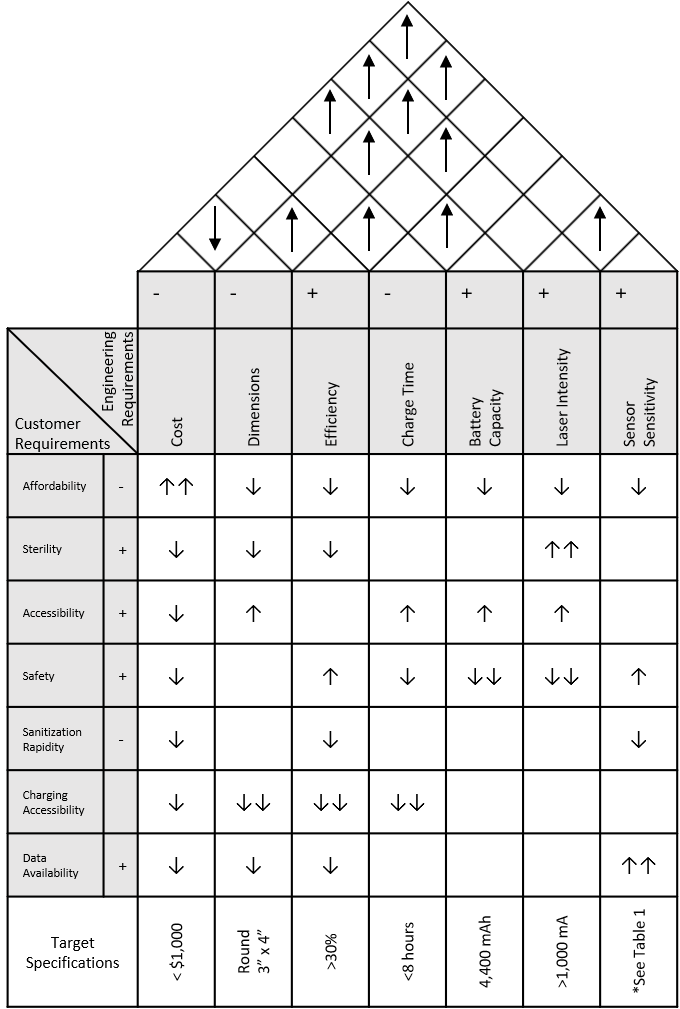


Figure 14: House of Quality

# Technology Investigation for Project Development

## Device Power

### Inductive Charging

The simplest way to implement a wireless charging system was to use technology that relies on Faraday’s Law to create a changing magnetic field in one coil, which in turn induces an AC voltage in a magnetically coupled coil nearby. Reducing the cost of the final device was one of the main engineering and marketing goals of this project, so it was strongly recommended to implement a wireless charging system that did not require a harmonic oscillator in the power transmitter. The electric power that is available through a standard household, single-phase socket already provides an alternating current at 60 Hertz that can be used for simple inductive charging.

There are three physical constraints that needed to be addressed in order to implement inductive charging well. All of them involved the coupling of the transmitter and receiver coils [6]. The first constraint for this type of charging is the distance between the two coils being used. In order to maximize the magnetic coupling between the transmitter and the receiver, it is necessary to minimize the distance between concentrically aligned coils. Part of this issue could have been taken care of by adding permanent magnets to the casing of the transmitter and receiver so that they are as close as physically possible during the charging process, but it was also possible to implement a simple clip that would keep the power transmitter in place. The other feature within our control was to minimize the thickness of the casing for the transmitter and receiver. Ideally, the distance between coils would be less than 10 millimeters, which was more than achieved by the final product's ~5 mm separation [6]. Along these same lines, it was necessary for the transmitter and receiver to actually be aligned concentrically in order to minimize the magnetic flux leakage between coils. This problem would also have been dealt with either by implementing some sort of clip-on device or by using permanent magnets fixed to the inside of the casing for the water bottle top and the power transmitter. In our design, we used electrical tape to produce a thin, effective harness attaching the receiver to the inside of the cap.

The third physical constraint for using inductive wireless charging was needing to minimize the surface area of conductive material between the two coils. Eddy currents are created in any conductive surface that has a magnetic flux passing through is perpendicularly. Adding permanent magnets into the centers of the transmitter and receiver coils inevitably creates eddy current losses which not only reduce the overall efficiency of the charging system but are also a considerable source of heat depending on the magnets used. These eddy currents could also be induced in the casing of either the bottle cap top or the power transmitter if either were made of some conductive material such as metal [7]. As there was no advantage to using a metal casing when compared to plastic, we chose to use a PLA cap with mounting screws located well away from the receiver, increasing the efficiency of the wireless charging. As for installing a magnet, although we would theoretically have been able to charge a 4,400 mAh battery within eight hours despite the losses in the magnets, it was very important to minimize how much heat was being produced within the water bottle cap since the intention was for the casing to eventually be sealed/sealable so as not to allow water ingress into the electronics.

The current standard in the market for inductive charging is called Qi (pronounced “Chee”) and this is specified by the Wireless Power Consortium [8]. Having a device that is Qi certified is much more complicated than simply sending and receiving power through electromagnetic induction, because this standard also specifies a communication protocol for data transfer between transmitter and receiver. This communication is used to provide feedback to the power transmitter (or base-station) about information such as the state-of-charge of the battery. Providing a means for this communication was a very important concern for charging a battery, because the worst-case scenario for continuing to push charge into a fully charged battery is starting a fire. There are premade Qi compliant power transmitters and receivers, which run for over $40 for the pair. This was much more expensive than the initial estimates for wireless charging modules but having a Qi compliant transmitter and receiver made the final product much safer as well as allowed for USB compatibility for the power transmitter.

### Magnetic Resonance Charging

The newer form of wireless charging technology that is available today relies on the concept of magnetic resonance. This design of wireless power transmission is still possible due to the electromagnetic induction of an AC voltage in a coil, but the main difference is that tuning the transmitter and receiver to oscillate at the same resonant frequency strongly supports the magnetic coupling between the two coils. In other words, the transmitter and receiver do not suffer so greatly from the coils being either misaligned concentrically or separated by distances much greater than those in which simple inductive charging can operate [6]. This feature was not thought to be as useful for the purposes of this project, because the intention was to have a power transmitter that sits on top of the water bottle and easily keep coils aligned and nearby. It is also important to note that even though this technology allows for coils to be further apart and misaligned that it still does not achieve efficiency levels as high as traditional inductive charging due to the flux leakage between coils. Although it would have been possible to design a circuit to use the power of magnetic resonance while still keeping the coils tightly coupled in space, the added drawbacks for this sort of design are that the oscillating circuits are more complex and that these resonant frequencies are typically much higher than those used in standard inductive charging. Introducing higher frequency power signals has the possibility of creating electromagnetic interference in nearby devices. For the purposes of this project, it seemed inadvisable to utilize this more advanced and newer technology.

### Battery

Upon investigating various popular vendors online, it became clear that the choice in battery chemical composition for the needs of this project was much more dependent upon availability than the specific differences in performance. The majority of battery packs that do not exceed three inches in any dimension are either lithium-ion (Li-ion) or lithium-polymer (Li-po) based. The handful of Nickel-Cadmium batteries that were found online were larger and heavier than the lithium-based counterparts of equivalent capacity as well as being more expensive in terms of charge capacity per dollar. Bearing this in mind, the two main aspects of design focus for the battery will be the rated charge capacity and safety circuitry. The options for Li-ion and Li-po battery packs that were investigated were typically rated at 3.7V/4.2V, which meant that the nominal rated voltage is 3.7 volts, and the maximum voltage of the battery is 4.2 volts [9]. It is typical that batteries will have a range of voltages that depends on what percentage of charge it has at the time, and this has many consequences in terms of additional circuitry that is needed to properly charge and discharge the units. One such consideration was designing voltage regulators to specifically meet the voltage and power requirements of each instrument in the project. Another factor that was more directly related to the battery itself is the need for special circuitry in ensure safely charging the battery packs. Having batteries with a denser energy content is naturally going to make them more dangerous, which is why so much care was taken to ensure safe implementation in this project.

Monitoring the voltage and current flowing in or out of a Li-ion/Li-po battery cell is typically handled by protection circuitry that is included in the battery itself [9]. However, even with these safety features installed in the battery packs, it is still highly recommended to use a charger that is specifically designed for Li-ion/Li-po batteries of the specified voltage. This additional circuitry, which needed to be purchased separately, is designed to help control the flow of current into the battery as it is charged. There are many nuances to charging these lithium-based batteries in a way that is safe, fast, and prolongs the life of the battery by not damaging it [10]. The main takeaway from this technology research was that the additional cost of professionally designed and manufactured safety circuitry seemed to be well worth the price, especially when considering the consequences of a Li-ion/Li-po battery that has been mishandled.

One final consideration on the topic of battery safety was monitoring the temperature of the cells. The major causes of batteries getting too hot that can be controlled is due to improper charging and discharging [10]. However, the intention of this project was to build a unit that does not allow the flow of air and water into the casing that contains the electronic circuitry and therefore it was not unreasonable to assume that the temperature of the battery pack could reach unacceptable levels due to a lack of proper heat dissipation during operation. A significant production cost of this project was already forecasted to consist of pre-made units to ensure proper wireless charging of the internal battery, but for the sake of safety and completion, it was thought to be worthwhile to also investigate the potential implementation of a temperature monitoring system.

### Supply Voltage Regulators

Regardless of what the actual voltage needs were for any of the components that were included in this project, they all required a consistent supply voltage that does not change as the battery changes temperature or its level of charge. There are prefabricated options available on the market as well as online tools for assisting in developing voltage regulators that were implemented in the printed circuit board and both of these options were explored.

#### LDO Linear Regulators

Low-Dropout (LDO) linear regulators are devices that are built specifically to provide voltage regulation for output voltages that are close in magnitude to the input voltage being provided [11]. Linear regulators do not rely on switching and therefore produce less noise in their output than the typical switching voltage regulators. However, the drawback of a linear regulator is that there is a minimum threshold voltage drop that must exist between the input and output in order to remain stable [11]. This minimum voltage drop that must be established over the internal transistor of a linear regulator is known as the dropout voltage; this can be a liability for a regulator when the input voltage is constantly changing, due to the nature of a battery, and the required output voltage is near the potential range of voltages that the battery provides. Not all linear regulators are designed to minimize the dropout voltage, but this was seen to be a highly desirable trait since it was originally not clear what actual output voltages would needed to be maintained.

#### Switching Voltage Regulators

There are several different topologies for switch-mode DC-DC converters that can either be used for stepping up and/or stepping down voltages. One of the main advantages of these switching voltage regulators is the relatively low amount of power that is lost in the switching components, which translates to having a device with a very high efficiency [12]. This higher efficiency was desirable, as it is one of the engineering requirements that has been outlined in the project description. There is a convenient online tool called WEBENCH Power Designer that is made available by Texas Instruments; this was how the switch-mode DC-DC converters that were required in the pre-requisites for this class were created. Using this online tool allowed for easily selecting between different designs based upon the total cost of components, switching frequency, design footprint, and many other variables of interest when designing a regulator to be implemented on a custom PCB.

It is worth specifically mentioning that it was highly desirable to minimize the Electromagnetic Interference (EMI) that would be created from implementing switch-mode voltage regulators due to the fact that we used Bluetooth wireless communication on the same board. Designing these types of DC-DC regulators with the added circuitry for filtering switching noise that causes EMI would have definitely made this the more expensive option when compared to the linear voltage regulators. It was not actually necessary to include additional filtering circuitry within the power system design due to the difference in frequency of the switching when compared to the frequency of the switching regulators. When some preemptive designs were created on WEBENCH, it was clear that the highest frequency of switching was in the 1-2 MHz range, while other options had switching frequencies closer to 500 kHz. Since the frequency used for Bluetooth communications is 2.4 GHz [13], the difference in these frequencies is at least four orders of magnitude. Since higher order harmonics of a signal occur at each integer multiple of the fundamental frequency, it could be inferred that any potential harmonics of the switching regulators that comes close to 2.4 GHz would have been negligible due to this difference in frequency magnitudes. Regardless of the potential differences in price, they were not so large as to offset the benefit of having much higher efficiency.

Not only does a higher efficiency mean longer battery life of the device, but it also means less heat being generated within the device. None of the members of this team had any formal experience in properly designing a system by calculating the maximum heat output, the heat dissipation, and then comparing these figures to the maximum safe temperature of the electronic components. Therefore, it seemed to be prudent that every possible effort be made to minimize the heat losses during power transmission and supply because the bottle cap was intended to be sealed, so as to prevent the ingress of moisture. Including a Lithium battery within this confined space will mean that every effort need to be made to ensure that it did not get too hot either during charging or discharging.

#### Current Regulating Diodes

This project had originally appeared to be requiring a lot of various regulated voltages. It was also clear that the microcontroller needed to be regulated to 3.0 V for device operation; however, there were other options for regulating device components. One option was implementing current regulating diodes (CRDs) for the optical components. They are also known as current limiting diodes or constant current diodes. This regulating circuit is typically composed of a JFET and is rated for a certain amount of current. Current regulating diodes that match the desired V-I graph for the optical diodes could have been purchased and implemented into this project. This could have required fewer regulators on the device, which overall could have decreased the cost of the project. As long as the supply voltage is high enough to support the voltage drop of each optical component (with some leeway), this could have essentially created miniature regulators for each optical diode to achieve their own regulated current and forward voltage. However, they technically only control a maximum current on the node; in other words, a current below the rating could have still occurred based on the voltage being supplied to the diode.

## Sanitization

How do you feel when you can see, taste, or smell a contaminant? It is bad, right? What do you think of the water that smells, looks, and tastes just fine? Is it good enough to drink? The answer to this is -- not necessarily. The water can be contaminated without having an odd flavor.

Microbial and organic substances cannot generally be recognized by human senses and can lead to severe health issues down the road- for microbes in particular, the affect is generally gradual enough as to allow for significant transmission prior to diagnosis. Water may contain contaminants from pesticide or compost application. These chemicals from pesticides and manures in “water may increase cancer risk and reproductive problems, and can impair eye, liver, kidney, and other body functions” [14].

The goal of this bottle is to sanitize the water in the bottle by killing the microbiological contaminants present in it. “The transmission of diseases such as typhoid and paratyphoid fevers, cholera, salmonellosis, and shigellosis can be controlled with treatments that substantially reduce the total number of viable microorganisms in the water” and FLASC does that by killing the pathogens by UVC light [2].

The goal of the sanitization component is to disinfect the tap water by elimination of microorganisms which are responsible for various waterborne diseases. There are various kinds of methods for the sanitization of water, the most common and widely used are chlorination, boiling and UVC light.

Choosing the method for the disinfection of water depends on various factors. These include:

* How effective it is in eliminating the microorganisms (bacteria, protozoa, viruses, and helminths).
* How reliable and accurate of the process and the way it can be controlled and monitored.
* Whether the method leaves some residuals behind and how would that affect the disinfection process and be taken care of.
* How purified the water gets after the process is complete; and
* How accessible the technology is for the public water supplies [2].

Now we will look into different sanitization methods used in the industry and compare them to see why we went with our design.

### Sanitization Methods

There are different kinds of methods used throughout the water irrigation industry for cleaning and purifying the water. The most common and widely used methods are boiling and chemical treatments. Ultraviolet light industry is rapidly growing in today’s world. The methods depend a lot on what scale the industry and purifying is happening and then the methods are considered. Some big industries use one, two or all the three methods for sanitizing the water. In the following section we will discuss more about each water treatments and compare them to get the best method for our design project.

#### Boiling System

This is one of the most common method used in a lot of areas throughout the world let it be how remote it is. In this method the source does not depend as it is ensured that even regardless of the source the water kills the pathogens on boiling. Boiling time of water depends on the altitude. Higher altitudes require longer time and lower altitudes require shorter time. “Water should be at a rolling boil for 1 minute and at altitudes greater than 6,564 feet, boil water for 3 minutes” [15]. This method has great advantages such as the source does not matter and requires a short amount of time to boil the water. But, for this project this method to kill the contaminants will not be ideal. As our product is a water bottle it would be difficult to incorporate the electronic system as it will require high voltage and hence will need a power source outside where it can be plugged to. The temperature is going to rise due to boiling and hence the material the bottle will be made of will have to be taken care of as the material should not disintegrate or mold at high temperatures. The heating will also result in the water to be too hot to drink and might also heat up the outer casing of the bottle for the user to hold it. We can add a cooling system to cool the water after the boiling process is completed but it will require more electronics. Even though these problems can be overcome it would not be the most suitable for the water bottle.

#### Chemical Treatments

There are several chemicals that can be used to disinfect the water. The most commonly used chemicals are chlorine, iodine, bromine, and oxidizing agents. Chlorination is a widely used process in big factories, but in a project like this it has various drawbacks as it leaves residuals behind, which can be dangerous when consumed. Other chemical like Iodine has a better efficiency but it is tough to store Iodine as it deteriorates in sunlight. It also leaves a bad metallic taste behind. Bromine has its drawbacks in storing and transporting too as it highly reactive. Bromine’s major drawback is its reactivity with ammonia or the other amines that affect its treatment of disinfecting the water. Oxidizing agents have great efficacy but due to its high cost and desired pH concentration it is not really viable for us to use it. All these being chemicals they can react and act differently with different foreign viruses and hence will not give the accuracy we need. This method is easily accessible and does not require any electrical components, but they have various drawbacks such as pH of water, temperature of water, it leaves residuals, leaves an after taste, and takes a longer time for disinfecting the water.

#### Ultraviolet Light

Ultraviolet (UV) light means “beyond violet” coming from the Latin word “ultra”. Naturally, UV light comes from the sun. UV light spectrum lies between visible light and x-rays. The discovery of UV light was done in 1801 by a German physicist Johann Wilhelm Ritter. He observed in his experiment that the light rays beyond violet light darkened silver slats [16]. Ultraviolet light has an electromagnetic spectrum ranging from 10 nm to 400 nm with corresponding frequency ranging from 750 THz to 30 PHz as shown in Figure 15 (see below). These wavelengths are too short for the human eyes to see.

Ultraviolet light is broken into four categories:

* Ultraviolet A (UVA) – also known as Long Wave/ Near UV as it has longer wavelength and is closer to the violet light. The wavelength ranges from 315 nm to 400 nm. This light is not blocked by the ozone layer.
* Ultraviolet B (UVB) – this is also known as the medium wave UV. It has a wavelength between 280 nm and 315 nm. This light not completely blocked by the ozone layer but most of it is absorbed by it.
* Ultraviolet C (UVC) – known as short wave UV or commonly referred to as germicidal. This light is completely absorbed by the ozone layer. UVC has wavelength between 100 nm to 280 nm.
* Vacuum Ultraviolet – this has a spectrum from 10 nm to 100 nm. They are absorbed by the nitrogen in the atmosphere and hence cannot penetrate through.

Electromagnetic Spectrum showing the wavelengths of sub parts of UV Light

Diagram

Description automatically generated

Figure 15: Electromagnetic Spectrum Figure taken from [17]

Pathogens like bacteria have genetic information coded like DNA and RNA similar to that of the cells in our body. We all know the steps for the central dogma of biology: DNA to mRNA to Proteins. If the process is interrupted at any time the cell dies. “DNA is like a blueprint; a little alteration can greatly affect the intended structure and cause a collapse in the entire cell” [18]. All viruses contain either DNA or RNA.

Sufficient intensity of any radiation has the possibility to kill. For example, when there is an atom bomb blast, the gamma radiation has the capability to vaporize an entire person within a few hundred feet but occasional photons of gamma radiation, which are experienced by all the passengers traveling by air causes no damage [19].

Now we know that occasional photons of gamma radiation can damage the DNA and RNA. When the damage done is minimal, cells have the ability to self-repair and recover. “The persistent radiation- induced changes in DNA/RNA may express themselves as cancer” [19]. When the radiation is strong then it completely damages the DNA/RNA, and the cell dies.

Similar to gamma radiation, UV light with appropriate wavelength and intensity can destroy the cells and viruses. DNA and RNA are particularly more sensitive to UVC light. The efficiency of destroying the microorganisms depends mainly on the distance, wavelength, intensity, and duration of exposure. “A standard UVC 270 nm LED fixture can kill most microbes within a six-inch radius in 10 seconds” [18]. DNA and RNA breakdown occurs due to ablation of the fatty acid molecules comprising the source code of life, as the high energy photons break off pieces. This molecular rearrangement of the biological components in the microorganisms causes inactivation of organisms functioning and hence kills it without leaving any residuals. The common pathogens found in ground water are Giardia, Cryptosporidium, and bacterial pathogens like E. Coli. UV radiations effectively inactivate these pathogens by interfering in their genetic information.

#### Methods Comparison

It is clear after looking into all three sanitization methods from Table 13 (see below) that ultraviolet light is the best approach to go forward with, due to its easy use and portability so that is what we implemented. Boiling system and the chemical treatments were not feasible methods for our product due to their lack of portability and everyday use as our audience is travelers and hikers. UV light requires more safety precautions due to its ability to harm humans with potential exposure but that is just a minor issue and was taken care by our safety measures using a reed switch.

Sanitization Methods Comparison

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Boiling** | **Chemical Treatment** | **Ultraviolet Light** |
| **Effectiveness** | Very High | Very High | 99.9% pure |
| **Time** | 1-3 min | 15-20 min | 10 sec – 2min |
| **Cost** | <$25 | <$20 | <$30 |
| **Residual Left** | None | Yes | None |
| **Electric Components feasibility** | Not feasible due to heating and cooling | No electronic components required | Feasible |
| **Implementation** | Hard | Hard | Medium |
| **Power** | High | Low | Medium |
| **Safety** | Medium- can get too heated | Medium – because of the residuals left | Medium – UV light can be harmful for humans |
| **User friendly** | Low – water gets too heated | Low – User needs to keep refilling the chemicals | High – no user interaction needed |
| **Portability** | Hard | Medium | Easy |

Table 13: Sanitization Methods Comparison

### Safety and Precautions of Ultraviolet Light

We all know and have heard about UV light not being good for our health in the long run and it is completely true. Most of the UV light is blocked by the ozone layer keeping us safe but now as the industry has started building UVC lights for its usefulness for killing microorganisms we need to be careful and take the necessary precautions. Following are the risks caused by UVC radiation and they all depend on the wavelength, duration of exposure and dose:

* It affects the skin and eye and can cause painful eye injuries and skin burns.
* Some UVC lamps generate ozone. “Ozone inhalation can be irritating to the airway” [20]. These ozone gases can cause several health complications like coughing, chest pain, lung damage, throat irritations, asthma, and more.
* As we know it interrupts the DNA and RNA in the cells and same could happen in human cells and cause many health issues. If UV light is not exposed correctly, it might just modify the virus and hence causing mutations of the virus that might be difficult to kill later.
* “High-exposure to radiation promotes the formation of cancerous tumors that can prove fatal if not detected in time” [18].

Therefore, proper safety precautions are very necessary when using a UVC light. Do not come in direct exposure to it and never look right into the path of it.

For these safety concerns we have a reed switch in the bottle that will not let the UVC light radiate when the cap is open for the safety purpose of the user. UVC light will only work when the bottle cap is sealed.

### Different types of UV Sources

Now as our group has decided that the most feasible method for our project is UV light, lets dive deeper into it. UV light has many different sources, and we had to decide on the best one for our project.

All UVC lamps are not the same. Some lamps may emit specific wavelengths of UV light (254 nm or 222nm), and some may emit broad spectrum of UV wavelengths. These lamps may be combined with emitting visible and infrared radiations too. The emitted wavelengths play a vital role in the effectiveness of the lamps and the health and safety risks associated with it. “There is some evidence that excimer lamps, with peak wavelength of 222-nm may cause less damage to the skin, eyes, and DNA than the 254 nm wavelength, but long-term safety data is lacking” [20]. Following are the different kinds of lamps used to emit UV light:

* Low-pressure mercury lamp: These are the most common lamps which are used in most of the water irrigation systems to disinfect the pathogens. Its main emission is at a specific wavelength of 254 nm. They can also produce other wavelengths, but this is the main one for UVC disinfection.
* Excimer lamp or Far-UVC lamp: These lamps are close to the infrared radiation and has a peak wavelength at 222 nm.
* Pulsed xenon lamps: These lamps are mainly used in hospital settings to disinfect the air in the operating rooms or other spaces. They emit short pulse of broad spectrum which includes UV infrared and visible. These lamps can cause many health hazards and for safety purposes are mainly used when there are no humans in the desired space.
* Light-emitting Diodes (LEDs): Light-emitting diodes emitting UV light are booming in the industry and are becoming very common in the industry and commercial use. They are getting commonly available and hence the demand is increasing. LEDs emits very narrow band of spectrum. There are multiple LEDs available in the market right now with different peak wavelengths like 265 nm, 275 nm, 280 nm, and others.

Comparison between UV LED and UV Lamp

Originally most of the disinfection systems for water to kill the microorganisms used UV lamps. UV lamps had a single wavelength light emitting at 254nm. Due to the increase in the technology most of the companies were looking into polychromatic lamps (meaning the light is emitted in multiple wavelengths) due to its better efficiency in killing the pathogens. Therefore, after referring to Table 14 (see below), the most feasible out of these options due to our design’s portability, size, energy, and cost constraint was to be using UV LEDs so we went with that.

LED Comparison

|  |  |  |
| --- | --- | --- |
| **UV LED** | **Section** | **UV Lamp** |
| New, Light, Simple, Compact, Small | Technology | Old, Bulky, Heavy, Complex, Large |
| 10,000 – 50,000 Hour | Lifetime | 2,000-10,000 Hour |
| Low | Energy Consumption | High |
| Zero | Warm-up time | Slow |
| No mercury, No Ozone | Environmentally Friendly | Mercury used, Ozone generation |
| Low | Heat Generation | High |
| Single UV Band, Customizable | Emission Wavelength | Multiple Peaks |
| None | Heavy Metal | Mercury (20 - 200 mg) |

Table 14: Comparison between UV LED and UV Lamp. Referenced from 21.

### LED

LED is a semiconductor device most commonly constructed out of a single P-N junction. It emits light when current is passed through it. The basic principle on which LED is works is that of a semiconductor. LEDs are made of p-type semiconductor placed on the top and a n-type semiconductor placed on the bottom. The lack and excess of electrons in the materials creates junction between them where electrons are passed when there is a current flow. When forward bias voltage is applied, holes and electrons move. Electrons from the n-type material are pushed to p-type material and holes from p-type material are pushed to the n-type material. At the depletion region or the junction between the two materials these holes and electrons combine and hence produce quantum energy due to the radiative recombination. This quantum energy can produce other energy like heat too, but with proper schematic of the semiconductor one can design it in a way to emit certain wavelength of light. Therefore, the material, quantum wells and dislocation density play a vital role in the semiconductors to emit specific wavelengths.

LEDs are available in many different wavelengths. They were originally used in infrared spectrum and then into visible light starting with red and green and then to blue and white. To produce LEDs in UV light was very difficult in historic times, but now we can see the tremendous growth in it due to the newer technology.

“The first UV-C LED devices were developed primarily in Japan, Korea and the US, as extensions to LED devices emitting in the blue and then UV-A and UV-B wavelength region” [22]. UV-LED are made from III-V semiconductors materials. The ratios of atoms from both the materials matters a lot. They are precisely chosen to fulfil the needs. The atomic number plays role in the energy bandgap and this in turn determines the wavelength emitted by the material. By alloying the material, we can alter and change the bandgap and hence the wavelength, which is very beneficial in the semiconductor industry. The commercial LEDs present in the market are mostly made of sapphire substrate with buffer layers of aluminum nitride (AlN) or aluminum gallium nitride (AlGaN). These films have more defects. “Any defect in the crystal structure is a point where electrical carriers can recombine in a way that does not produce the desired wavelength and, thus, the efficiency is reduced” [23]. Growing and using Aluminum Nitride substrate is defect free and hence more reliable for UV LEDs. The shortest wavelength that was emitted using aluminum nitride substrate with very little dopant was 210 nm. Therefore, most commercial UV LEDs are made of AlN and AlGaN with some dopants. The efficiency decreases as the emitted wavelength decreases due to the material lability. These LEDs are very reliant and therefore benefits our project. “A test of 170 UV-C-LEDs fabricated over the course of six months showed a median degradation of less than 4% after 1,000 hours of continuous operation at 100 mA,” this shows the durability too, which is very important for our design [23]. To choose the correct LED was a tough task for our project design as we want it to be cost efficient. Most of the UV LEDs are expensive as they are very difficult to make. AlGaN is also tough to make and hence there are only a few reliable and efficient semiconductor industries making these materials efficiently. The special ceramic packing, quartz window and the chip used makes it more complicated and expensive. The chip’s structure makes it difficult to optimize it. Therefore, selecting the LEDs to be implemented in our project required a lot of research and finding which led us to the desired LEDs discussed in a later section, which fit our requirements perfectly.

### Technology in the Industry

Water is everything for human beings. Water keeps us hydrated, which results in a better metabolism and better skin. Recently with the decrease in the amount of natural clean drinkable water, water purification industry is at its rise to provide clean and safe drinking water for every individual all around the world. Also, with the covid-19 pandemic, UV-C light technology has become more popular and boomed as it kills the viruses and bacteria.

There are different technologies being implemented everywhere for sanitizing the water and air. All these technologies are great and have been proven to be very efficient. Figure 16 shows some of the technologies used in the water disinfecting industry to purify the water. The first one shows a filter and purifier attached to the plumbing, so the water passes through it and gets cleaned and is then transferred to the output source using chlorination and filters. The second image showing Pearlaqua is a UV-C LED purifier which is small and compact so it can fit in anywhere and can have multiple use. It has different features like intensity monitoring and temperature independent, which are useful to know about when sanitizing water using UV-C. It could be placed between any two pipes and can clean the water as it is passed through it. The third image shows Acuva, which is a large container with multiple UV-C LEDs. It is also connected to the pipes and water is allowed to let flow through it to purify it. Major big industries have everything combined like chemical treatment, boiling and UV sanitization to get purified water. They work on large samples and processes and hence need that, but for our households and daily use these small systems have proven to be very reliable and efficient for healthy drinking water.

Commercially Available Sanitizers

**A picture containing graphical user interface

Description automatically generated**

Figure 16. Showing some of the examples of water disinfectant in the industry. Figure taken from [22].

Main differences for our household and everyday use of water filters and sanitizers from the industry are implementation and use of different filters and UVC intensity and efficiency. There can be purifiers with multiple LEDs or single LEDs. Both are very efficient but have their pros and cons. Figure 17 shows a LED array sanitizer used in industrial settings. This device, produced by the Universal Science company, is called 'SpecialRED.' It has a strip of LEDs on the side and a slot in which a clear bottle can be placed in the handle or other things you want to disinfect, and it will illuminate the light and sanitize them. Other devices consist of multiple UV LEDs, and we can just place a bottle on the top and it can sanitize the water from different angles. The purpose of using multiple LEDs is for greater intensity of light, which results in more efficiency. Also, if one LED fails and does not output enough intensity there are other LEDs to take care of it and still purify the water. The disadvantages are it uses a lot of power and hence more electronics and higher voltages battery to supply enough current and voltage, requires a bigger space therefore not handy and easily portable and carried to different places and gets more expensive as it uses multiple LEDs, which are not cheap as discussed in the LED section.

Commercial LED Sanitizer

Diagram

Description automatically generated

Figure 17: SpecialRED Sanitizer. Figure taken from [24].

Figure 18 shows a single semiconductor LED in different packaging styles. Part (a) is TO packaging which was used in the earlier days before the semiconductor (SMD) packaging was invented, (b) shows typical semiconductor packaging LED, which is mostly used in today’s industry and (c) shows how the SMD LED can be incorporated on a PCB board on the circuit to be used. These LEDs can be placed anywhere in the system and does not require many electronics or optics and hence it is easily compatible to any use. They are very small and easily accessible. They are comparatively cheap as they are single piece. Mostly all the UV-C water bottles in the industry right now use the PCB LED and these bottles have proven to be very efficient with their effectiveness in sanitizing the water. There are some disadvantages to it like if the LED does not produce enough intensity, then the water is not sanitized completely and could leave to false interpretation, therefore the system needs to take that into account too. Choosing the LED for the system needs different analysis and we need to check the specifications and the datasheet correctly as the LED needs to have enough output power to purify the water, with appropriate wavelength, as not all the LEDs can have the required specifications needed for our specific design.

Commercially Available LEDs

A hand holding a key

Description automatically generated with medium confidenceA black video game controller

Description automatically generated with medium confidence

Figure 18. Single LED.

1. LED with TO packaging, (b) Semiconductor packaging LED. Figure (a) and (b) taken from [22] (b) LED placed on the PCB board to be used in the system. Figure taken from [24]

Therefore, after the analysis, our group chose to go with using 2 single LEDs as it was cost efficient, small to be fit anywhere like the bottle cap, easy to implement and did not require a lot of electrical power to function them and hence keeping the system small and compact. This fit perfectly with our design and the requirements for the project.

### Tunable UV spectrum and it’s efficiency

Most of the UV light emitted by the UV lamps and LEDs used for water purification are single wavelength. After increasing demand of UV and technology, the interest in UV lamps increased for polychromatic lamps, meaning the wavelength is emitted in different wavelengths. In 2011 as a part of Water Research Foundation, National Institute of Standards and Technology (NIST) was asked to help build a tunable UV laser for irradiating water samples [26]. NIST build a tunable laser of wavelength ranging from 210 nm to 300 nm. The foundation did not want to use lamps as they have broad bandwidths, which causes difficulty in testing the specific wavelengths efficiency on different pathogens. Even using the filters did not help that much and still unwanted frequencies were passed through and acted on the sample. Therefore, NIST made a tunable laser with very narrow bandwidth of about 1 nm so it could be tuned with precision to check the efficiency of each wavelength in the spectrum. “By tuning the laser wavelength and controlling the dose exposure, an action spectrum or germicidal effectiveness curve is developed to quantify the efficiency of micro-organism inactivation” [26]. The action spectrum is defined as “a plot of a relative biological or chemical photoresponse (= Δ*y*) per number of incident (prior to absorption) photons, vs wavelength” [27]. Figure 19 shows the DNA damage and the infectivity caused at different wavelengths in the UV spectrum. From this we can see that at wavelengths below 240 nm there is a higher absorption of UV by the proteins and at 260 nm the loss of viral infectivity is lower than the DNA damage and at 254 nm, 270 nm, 280 nm, and 290 nm there is no difference in the infectivity and DNA damage. Therefore, it is evident from the figure that wavelengths below 240 nm have a greater difference between the genome damage and rate of inactivation, whereas it is relatively similar at wavelengths above 240 nm. Comparing rate of inactivation at 210 nm and 254 nm we can see that it is almost 16 times greater at 210 nm and then it decreases rapidly. Most of the biological pathogens follow this trend, with a little difference amongst them. This shows that DNA damage is not the only thing responsible for inactivation of viruses in water using a range of wavelengths. The low wavelengths show an important role in inactivating the pathogens and hence can be tailored in a way to combine with other wavelengths to get the best result in purifying water in the industry.

Spectral Sensitivity of Adenoviruses

**Diagram

Description automatically generated**

Figure 19. Spectral sensitivity of adenovirus 2 and its DNA damage. Figure from [28].

### Selecting the LEDs

From all the knowledge discussed earlier the best option to pick LEDs are picking a semiconductor LED made from materials AlGaN, AlN or InGaN and choosing a couple of LEDs with different wavelength so we get a broad spectrum. In choosing the LED, we had to keep in consideration the size, cost, and energy constraint. The LED also needed to produce a good amount of intensity at the given wavelength to purify the water effectively.

#### M275D3

This LED is from Thorlabs, which is a very trusted company and therefore the assurance of the product would be great, and the product will be very reliable. The peak wavelength of this LED is 275 nm. The LED comes with a PCB and hence easy to use and implement in the product. The PCB is metal core and made in a way that it will seek in the heat emitted by the LED and hence protect the system and the product. The LED requires 12 V forward voltage and 300 mA maximum current. The output power generated by this is typically 68.3 mW. The emitter size is 2.7 mm x 3.3 mm, and the PCB thickness is 1.6 mm. The cost of this product is $139.67.

#### SU CULCN1.VC OSLON® UV 3636

This product is from Osram. This is also a well-known and well trusted company. The LED is made from AlGaN based flip chip and ceramic packaging. This is only the LED and does not have a PCB board with it, so it is difficult to implement it as it is very tiny and hence difficult to solder and implement it. The size of the LED is 3.5 mm x 3.5 mm x 1.71 mm. The peak wavelength of this LED is 275 nm. Voltage and current required by this LED are typically 5.7 V and 100 mA. The power generated is 10 - 13 mW. The cost for this LED is $6.27.

#### 3535 LED Diode

MPN: lAf1186826  
Manufacturer Part Number: gfW8379622

This product is from eBay made from a company in China. There are no details on the company and hence it does not seem that reliable, but it has got good reviews and rating, so it probably is kind of trustworthy. The peak wavelength is 265 nm. The voltage and current required is 5-6 V and 100 mA. The power generated is 2-5 mW. This product comes with a copper CREE PCB and hence easy to implement and test the product by just soldering wires at positive and negative ends and connecting to a power source with appropriate voltage and current. The size of this LED with the PCB is 10 mm in the diameter. The cost of this LED is $4.89. They also have a similar one for 275 nm on eBay which costs $4.30.

#### UVC 250-260NM

This LED is from Shenzhen Trillion Auspicious Lighting Co., Ltd. This is a new company and they do not have those many reviews yet. The peak wavelength is 256.4 nm. The LED requires 24 V voltage and has a power consumption of 3 W. The radiant flux is 120 – 200 mW. The cost of this LED is $18.50.

#### DS-UV254P3Z-3535AC-S-06

This LED is from Zhejiang Danse Electronic Technology Co., Ltd This is also a new company. The peak wavelength is 255 nm. The LED requires a voltage of 5-7 V and current of 100 mA. The give out power of 10 -12 mW. This does not come with PCB and hence tough to implement and get the LED working as it is very tiny and has difficulties in soldering or finding the perfect mount for it. This LED is made of AlGaInP material. the cost of this LED is $3.85

#### HOUKEM-02W-255

This LED is from Dongguan Houke Electronic Co., Ltd. They are a new company from China too. This LED is made from InGaN. The peak wavelength is 255 nm. This requires 5.5 – 6 V voltage and 20 mA current. Typical output optical power is 3.5 mW. Not really sure if this comes with a PCB or not, but most likely it does not. The cost of this LED is $2.00.

After going through all these possibilities, we narrowed down to only two LEDs which matched our product design and requirements. We had to look for size, material, cost, power, and electrical requirements. Due to these constraints the 3535 LED Diode from eBay proved to be the best match as they were cheap, easy to implement, small and did not require a lot of power to function. We chose both the LEDs with peak wavelengths of 265 nm and 275 nm as we have a broader spectrum and greater intensity.

### LED Design

Referring to Figure 9 design we had to test, implement, and modify the schematic for the optical path ray and went with Figure 10 design. Referring to Figure 9 design we had to test, implement, and modify the schematic for the optical path ray and went with Figure 10 design. Using a beam splitter, diffraction gratings or a dichroic mirror did not work for our system. The reason was that due to the very small bandwidth difference between the two LEDs, optical components like beam splitter, diffraction gratings, and dichroic mirror could not work efficiently to combine the beams. They require larger difference in bandwidth to combine them. The phase matching was very difficult in LEDs and when we used optical components to combine beams, they should have a big difference in the bandwidths so one of them passes through and the other one reflects, but here the bandwidth difference being negligible it was very difficult to do so, and both passed in the same direction. Therefore, for the most efficiency we just had two holes in the bottom of the cap design and without any interference let the LEDs project out the UV spectrum and sanitize the water. This worked great as we did not lose any power using mirrors and lenses and had greater intensity and efficiency.

### LED Feedback Mechanism

It is important to know if the LED is providing enough power and sanitizing the water to be safe to drink. The LEDs might lose the power, get burnt out or just die and the user might not know about it and would just trust the system and drink the water. For this reason, we needed a feedback mechanism which tell us if the LED is providing sufficient intensity and hence giving us the confirmation that the water after sanitization process is clean and safe to drink.

One way to get a feedback mechanism is by radiometry. Radiometry works on the principle of converting radiant energy to heat or electric current energy which can be used to test the system. There are several detectors which can be used in this based in the wavelengths, field of views, sensitivities, and other characteristics. To get the correct detector and have it calibrated to your needs one needs to define the exposure dose and spectral irradiance. There are different ways to measure the spectral irradiance like directly by using a spectroradiometer, by luminous flux and by human eye or by calculation from the radiance data. Important factors we need to take care of in choosing the detector which may affect the process are precision, sensitivity, accurate calibration, dynamic range – the upper and lower limit of the irradiance, field of view, cost, speed of response, and portability.

In markets today there are several detectors available for UV wavelengths such as pyroelectrics, photodiodes, photomultipliers, thermopiles, and photoresistors. These systems work on converting the photon energy to current or voltage and then measuring that system to get the results. Photomultipliers, vacuum photodiodes, pyroelectric detectors, and photoresistors convert radiant energy to current whereas solid- state photodiodes convert it to voltage and thermopiles convert heat energy to voltage. Photomultipliers have great sensitivity but require very high voltage and are very fragile. They also lose sensitivity with time due to the alteration of the photochemical used in photocathode material. Vacuum photodiodes are similar to the photomultipliers but are not as fragile as the photomultipliers and hence better for portability and ruggedness. Solid-state photodiodes depend a lot on the material used to make them. Everything depends on the band gap energy of the material. They are sensitive to temperature increase, and they have two different modes: photoconductive and photovoltaic based on forward biased or reverse biased voltage supply. Photoconductive (reverse- biased) mode is more sensitive. Pyroelectric detectors are great and have very high precision but are very expensive due to the materials used to make them. Thermopiles are not sensitive detectors and depend on temperature for accurate measurement. Photoresistors are not as sensitive as any of the above detectors discussed but are cheap. portable and easy to implement and use.

For our system the best options out of all of them were a solid-state photodiode, a photoresistor or a phototransistor. After testing we decided to go with using a photoresistor. We had to use a fluorescent post-it as a filter to capture the radiant energy. Due to our portability and size constraint this was a perfect choice. Photodiodes were more sensitive but also more expensive and needed more electronics. We just needed to know if the LED is working or no and a photoresistor can let us know. Therefore, analyzing all the factors the best fit for our project was to go with the photoresistor as our feedback to check if the LEDs are working properly or not.

## Water Clarity Analysis

This section covers both the obsolete spectrometer design (water quality analyzer) and final water clarity analyzer. For the purposes of this section, "water quality" is synonymous with the presence of contamination from chemical sources (e.g. fluoride).

### Methodology

Water quality/clarity can be analyzed by a variety of chemical, optical, and electrical means. For the purposes of this project chemical methods were an inherently poor choice due to their disposability- a water bottle that requires monthly refills of obscure compounds and potentially toxic solutes is a poor choice for a hydration vessel. This left optical, electrical, and opto-electrical methods as potentially feasible methods.

#### Infrared Spectrometry

Infrared spectrometry identifies materials by introducing a known, broad, infrared spectrum to the sample. The returned spectrum, having transmitted through the sample, is then compared to the original spectrum [29]. By identifying the absorption spectrum of the sample, the chemical composition can be deduced to a high degree of accuracy (less than a ppm for gases) [29]. Infrared spectrometry has a shortcoming with aqueous solutions, however, as water is highly absorptive in the infrared regime. This quality makes infrared spectrometry a non-ideal solution for a spectrometer dedicated to the detection of chemicals dissolved in water.

#### Raman Spectroscopy

Raman spectroscopy appeared to be the most relevant technology for the obsolete water quality analyzer as it is a non-destructive, contactless optical method capable of detecting minute concentrations of dissolved materials. This methodology exploits a non-linear effect of light knows as Raman Scattering, which involves a small portion of incident light shifting from the characteristic wavelength due to interactions with the vibration of atomic bonds [29]. This method, while first discovered using the broad spectrum of sunlight, needs its pump source to be monochromatic to obtain usable data [29]. This method is non-destructive but is observed in the UV range, which necessitates special care for imaging sensors to avoid ablation [29]. While most usages involve near-infrared pumps, this method can use pump sources in the visible range to reduce absorption by aqueous solutions [30].

#### Fluorescence Spectrometry

Fluorescence spectrometry involves the stimulation of a sample with UV light, whose frequency-shifted response is shifted by contained fluorescent materials (primarily present in micro-organisms) [31]. This system, while experimentally demonstrated to be effective at determining the quantity and type of micro-organism present in a sample, is not of particular importance for this project’s design, as our product will destroy living contaminants. This methodology could potentially have been useful for demonstration purposes to provide a rapid means for showing the efficacy of the sanitization process, but rapid coliform testing proved effective enough.

#### Inductively Coupled Plasma

Inductively Coupled Plasma (ICP) is a method by which a sample is stabilized to a desired PH using bases or acids before introduction to a plasma source generated by electromagnetic inductors [32]. The process results in temperatures in the range of several thousand Kelvin while drawing multiple kilowatt-hours of electricity and destroying the sample- despite its effective and reliable spectral output, the system is neither low-cost, low-power, simple, nor compact [32].

#### Impedance Analysis

Impedance analysis detects and identifies the presence of solutes by registering the impedance among an array of electrodes of differing compositions [33]. The impedances are collectively compared to known, contaminated scores for similarity analysis- a similar response indicating the presence of the contaminant in question [33]. While impedance analysis is a reliable means of detecting dissolved heavy metals, the method requires submerged electrodes which would have been an undesirable departure from the project’s modularity (as the electrodes would need to be in the bottle rather than in the cap to ensure contact with the sample) while also introducing new surfaces that would have required sanitization and cleaning. Even under ideal circumstance these electrodes would undoubtedly have added a slight metallic flavor to the bottle contents- a process which would be accelerated by a UV sanitizer.

#### Summary of Methodologies

The methodology most applicable to this application was Raman spectroscopy, however, this was later abandoned due to equipment failures and repeated shipping delays in favor of a straightforward absorption analysis of 532 and 805 nm light. ICP would have required extremely high temperatures and currents while impedance necessitated submerged probes that were likely to introduce undesirable flavors. Fluorescence spectrometry, while it would have been unreliable for detecting the project's scope of contaminants, would have potentially been useful for demonstrating the effectiveness of the sanitization module during demonstrations (see Table 15 below).

Summary of Methodologies

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Infrared | Raman | Fluorescence | ICP | Impedance |
| Senses | Atoms | Atoms | Micro-organisms | Atoms | Atoms |
| Rapidity | Rapid | Rapid | Rapid | Slow | Rapid |
| Cost | Mid-range | Mid-range | Low-cost | High cost | Mid-range |
| Destructive | No | No | No | Yes | Partially |
| Feasibility | Unfeasible | Feasible | Feasible for demonstrations | Unfeasible | Feasible but undesirable |

Table 15: Summary of Methodologies

### Spectral Isolators (Obsolete)

While the function of this component would have been the same for both diffraction gratings and prisms, the differing implementation would have resulted in key differences for the overall system. The scope of this technological investigation was, in hindsight, too narrow, as technologies such as the Fabry-Perot etalon were not covered.

#### Prism

Prisms are the most basic means for separating incident light into its spectral components. Light of differing wavelengths is refracted at different angles in the prism due to the wavelength-dependence of the material’s refractive index. While prisms made from ice, plastic, candy, and resin are all usable in a high-school science demonstration, it is worth noting that many materials used to produce prisms absorb UV radiation and/or are outright destroyed by it, while tailor-made glass (such as THORLAB’s UV fused silica) is somewhat expensive [34]. This limitation would have been of particularly concern for traditional Raman spectroscope, which typically measure their outputs in the UV region.

#### Diffraction Grating

Diffraction gratings use grooves in a reflective or transparent sheet/panel to separate light into its spectral components [35]. While lighter and potentially smaller than prisms, diffraction gratings experience worse efficiency rates than prisms [34][35]. Additionally, the materials used to construct more economical gratings are vulnerable to ablation in intense UV.

#### Summary of Isolators

Neither isolator was a clear winner, and the spectrometer design switched between both components several times during the design process. The last design prior to the overhaul, however, used a prism to isolate the spectral components of the sample, as the pump laser had been changed to a visible wavelength making absorption in the UV schema a non-issue. The prism's correspondingly low cost for the visible regime (see Table 15 below) gave it the lead.

Summary of Isolators

|  |  |  |
| --- | --- | --- |
|  | Prism | Diffraction Grating |
| Cost | Low | High-Medium |
| Compactness | Moderate | Moderate |
| Efficiency | Moderate | Moderate-Low |

Table 16: Summary of Isolators

### Optical Detector

For the final design, this system utilized a stationary, singular phototransistor due to its excellent responsivity, easy integration, and low cost.

For the original design, a detector was required to convert the optical power in the analyzer into usable electrical signals, which would have been accomplished by shining the spectrally separated beam onto an array of detectors or by projecting the beam onto a single detector and shifting the spectrum across it. If the beam were separated in a horizontal manner, then the resultant image on the sensor would vary in wavelength based on its horizontal location.

#### CCD

Charge-coupled devices (CCDs) consist of an array of metal oxide semiconductors (MOS) in parallel to an array of illumination-shielded semiconductors [36]. CCDs operate by collecting optical energy and passing said energy as an electric charge into their respective semiconductor cell, which is shifted toward a readout cell every clock cycle. This results in a constant stream of individual cells outputting charge, which can be formed into an image.

#### CMOS

Complementary metal-oxide semiconductors (CMOS) are similar to their CCD predecessors but contain readout cells on every pixel [36]. This added complexity is beneficial for some purposes but until relatively recently was a substantial engineering hurdle resulting in increased cost, although manufacturing techniques have improved significantly. Essentially all commercially available CMOS (and CCD) sensors are packaged with a microchip that converts these readouts into a composite video signal, which effectively expresses information on color, intensity, and sound (from an onboard microphone, if applicable) for each pixel in a clever and complex manner [37]. This complexity, unfortunately, makes device integration with microcontrollers a titanic feat (if done without a base of functions) worthy of its own senior design project.

#### Photodiodes

Photodiodes are produced in single diodes or arrays of diodes. Photodiodes are the simplest of the researched technologies, consisting of a PN junction or an array of PN junctions constituting an array of pixels, all of which are continuously active [36][38]. These devices are attractive for spectrometers due to their large pixels and narrow gaps, reducing the quantity of light (and thus information) lost. Photodiode arrays are industrially available which are tailored to specific spectral ranges (such as UV or IR), which is generally lacking in commercially available CCD and CMOS chips [43].

#### Phototransistors

Phototransistors, like photodiodes, are semiconductor junctions which become more conductive under illumination [38]. Unlike photodiodes, however, phototransistors use a PIN junction to achieve higher responsivity (sensitivity) at the cost of a poorer frequency response [38]. This higher responsivity is also coupled with a loss in linearity, however- and while not important for a simple switching mechanism, this makes precise measurements with phototransistors a challenging feat [38].

#### Summary of Sensors

While a CMOS sensor was originally slated for the project due to its commercial availability and high resolution, the device's output signal is beyond the realm of feasibility for a cost-efficient microcontroller and would cause a significant increase in software complexity as well. In addition, the number of distinct values of intensity is rather low. Photodiodes were the original component-of-choice due to their linearity, low cost, and decent responsivity, but a phototransistor was utilized in the final design due to its superior responsivity and the lack of a need to compare readings at wildly different power levels, as would be the case in a spectrometer (see Table 17 below).

Summary of Sensors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | CCD | CMOS | Photodiodes | Phototransistors |
| Cost | Low-Medium | Low-Medium | Low-High | Low |
| Compactness | Medium-High | Medium-High | Low-High | High |
| Sensitivity | Medium | Medium | Medium | High |
| Ablation | Possible | Possible | Unlikely | Unlikely |

Table 17: Summary of Sensors

### Rotational Mechanism (Obsolete)

The spectrometer designs utilized a rotational mechanism to sweep the spectrum over the sensor. A servomotor or other (more) precise rotator was required- the most advanced version was intended to use a linear stepper motor.

#### DC Motors

Direct current motors are an excellent way to convert electricity to rotational movement, providing consistent torque regardless of resistance from the load. These motors, however, were utterly useless for applications requiring strict angular precision, as they move in a continuous fashion rather than incrementally. While a solution could have theoretically be attained by using pulses of consistent duration and current as an increment, and reducing the output with a high gear ratio, these remain excessively imprecise for this application.

#### Stepper Motors

Stepper motors "step" their axles by a set angular value in response to electronic input. These devices are capable of stepping in the clockwise and counterclockwise orientations based on the input signal and can eve rotate partial steps to achieve greater accuracies or smoother operation. This means that the internal gearing dictates the angular resolution of the component, with the step angle as the baseline off which all increments are made, unless a driver board with microstepping is used. For commercially available stepper motors of a size implementable in this project, the angular resolution tended to be 5.625° or 18°, corresponding to 64 and 20 teeth in the internal gearing, respectively [39][40]. While it was apparent that these angular resolutions were insufficient for the spectrometer, this could have been remedied by using external gearing or, as was implemented in the last spectrometer design, microstepping [41]. While external gearing would have added unwanted mechanical complexity to the design, it would have allowed for the resolution to be tailored to fit the project, as a modified gear ratio allows for higher angular resolutions.

Stepper motors have a disadvantage regarding heat accumulation, as they consume relatively large amounts of power to maintain position, especially when microstepping is utilized. This was poised to be an issue for the design, and the work-around was intended to be a heat pipe from the motor to the bottle.

#### Servomotors

Servomotors are, most directly, a more advanced version of a stepper motor. These devices are directed to specific positions using a PWM signal duration to dictate the exact angle [42]. This is generally accomplished by having a header duration (to signify that a signal is being sent) followed by a duration actually representing the position desired (for a servo with 180 degree rotation this could involve 1 or 2 millisecond signal durations for -90° and 90° respectively, with angles between determined by partial milliseconds [42]. This is especially advantageous in the event of an unexpected power failure, as servomotors can simply be reset to their starting position while stepper motors require more creative means of returning to a known position. Worth noting, however, is that the "starting position" was prone to change in the spectrometer, as variations in pressure exerted from on the FLASC and other forces may have adjusted the alignment of the system, and temperature changes will affect the characteristic wavelength of the laser diode as well. The angular resolution of these devices is generally not advertised on commercial datasheets for the hobby retailers investigated, and while the angular resolution isn't non-existent (as with normal DC motors), it may have been necessary to implement a gearbox with a servomotor as well. These devices accomplish their position-reading functions using clever but imperfect mechanisms (namely, the angular value is determined by measuring the resistance offered by a potentiometer, which serves as the final axle), leading to potential issues regarding accuracy.

#### Summary of Rotational Mechanisms

Based on the technological investigation (summarized below in Table 18), we believed that a linear stepper motor was the most desirable option due to the simplicity of implementation involved and good resolution with microstepping.

Summary of Rotational Mechanisms

|  |  |  |  |
| --- | --- | --- | --- |
|  | DC Motor | Stepper Motor | Servomotor |
| Cost | Low | Low | Low |
| Precision | Low | Excellent | Decent |
| Resolution | Low | Medium | Decent |
| Feasibility | Unfeasible | Feasible with a gearbox or as a linear stage | Feasible, may require a gearbox |

Table 18: Summary of Rotational Mechanisms

#### Summary of Analysis

Of the technologies available and feasible for the project, the most appealing design was to consist of a Raman spectroscope with a prism as a spectral isolator, a photodiode as a sensor, and a stepper motor for sweeping the spectrum across the photodiode. Component failure during testing, however, resulted in an analyzer which did not require a spectral isolator or rotational mechanism, instead utilizing only the laser diode and a phototransistor.

## Safety Switch Detection Device

It was of utmost importance to ensure that any lasers for either quality sensing or sanitization could be activated accidentally while the cap is detached from the bottle. Having a safety switch hardwired into the circuitry that can detect whether or not the cap is secured onto the bottle prevented the user from shining any harmful rays towards someone’s eyes or skin. It was greatly desired to implement a power safety switch that did not rely on physical pressure, due to the possibility of an inadvertent tripping of the switch due to pressure from either a user’s fingers or some surface that the bottle cap may be resting upon. Therefore, it was suggested that a device would be chosen that relies upon the detection of a constant magnetic field. Attaching a magnet to the water bottle towards the top allowed the safety switch to close while the lid is properly secured. It would certainly be possible for an intentional override of this safety mechanism by using another magnet but designing a product that is preventative of malicious abuse was outside of the scope of this project.

### Reed Switch

One device that relies on the presence of a magnetic field in order to engage is the Reed switch. The advantages of using this type of component were that they do not need added circuit components to be used and they do not constantly draw power [44]. However, it is important to note that a Reed switch consists of reeds enclosed in a glass container and this device is sensitive to intense and abrupt acceleration [44]. There were many sensitive optical components within the project device that do not do well if the cap is dropped, so it seemed to be a reasonable assumption that a Reed switch would not be the only component that was sensitive to sudden impact. It is also worth noting that a Reed switch would not be damaged by the introduction of moisture into its surroundings [44]. Although the intention was to properly seal the electronic components within the casing of the water bottle cap from the humidity of the bottle or its surroundings, it was an added bonus to have one more component that is resistant to water vapor.

### Hall Effect Switch

A Hall effect switch is a semiconductor device that can deliver a triggering output signal in the presence of magnetic fields [44]. Although these devices do need a constant supply of current for operation, it would have been definitely advantageous that Hall effect switches are solid-state devices without any moving elements [44]. This difference in device construction would have made the safety switch much more resistant to damage from sudden impact caused by the dropping of the water bottle cap. However, it has already been established that dropping the project device would already be extremely destructive to other vital components and therefore this physical robustness seemed to be a fairly unnecessary advantage.

### Safety Switch Type Selection

Overall, the lower complexity and power requirements of the Reed switches were the biggest selling points for this type of technology. Some of the main constraints for this project were the limited amount of energy afforded by whichever battery would have fit in the cap and the limited amount of space in which all of the components needed to fit together. Choosing a safety switch device that maximized the product battery life while minimizing the complexity of the printed circuit board definitely helped to create a more competitive alternative to the existing products of similar nature.

### Reed Switch Selection

Investigating the available options for Reed switches brought up a discussion about the best type of connection to incorporate the switch onto the PCB while also positioning it appropriately near the bottom of the cap casing. There are different magnetic sensitivities available for the Reed switches on the market but having a more sensitive switch that is closer to the top of the cap would have increased the chance that the magnetic flux from the wireless charging could accidentally actuate the Reed switch. Rather than using a through-hole or surface mount device, there were sensors in the market that come with wire leads installed onto the switch. Using a design of this type gave us much more flexibility as to where we adhere the Reed switch within the cap casing. In this way, we kept the safety switch as close to the magnet that was intended to activate it while maximizing the distance from the wireless charging portion of the device.

At this point, three options from the same manufacturer had been found that each had different sensitivities listed on the sales website. The three options that were considered are listed below in Table 19, and the only difference that seemed to be listed was the sensitivity, because the listed datasheet for each online is the same with a listed range of “pull in range available” being between 10 – 25 AT (Ampere-turns). These sensitivities are given in terms of magnetomotive force (MMF or F) which can be related to the listed magnetic flux density (B) of a magnet by B = µ\*F/l, where µ is the total magnetic permeability of the path between the two poles of the magnet and l is the length of that path (as learned in EEL 4205, Electric Machinery). As discussed in the wireless charging section, we were trying to use a cap material that has a high reluctance to magnetic fields so as not to interfere with the interactions between the desired electromagnetic components. Therefore, it was probably safest to assume that the magnetic permeability of the path between poles that would include the Reed switch while the cap is closed is relatively close to that of free space, µo = 4\*π\*10-7 N/A2 [45]. However, in making a calculation for the path for the magnetic field lines, it would have needed to be considered that these are concentric arcs due to the fact that the Reed switch needed to be coaxial with the magnet. Taking all of this into consideration, it was encouraged that we should simply buy the most and least sensitive options below so that we had flexibility in terms of the placement of the switch in the cap and the magnet on the bottle. Since each switch was only $2.03 and either would be reusable between different iterations of project design testing, it was not thought to be an undue expense to buy two different switches with different levels of sensitivity.

Summary of Reed Switch Options

|  |  |  |  |
| --- | --- | --- | --- |
|  | MS-2431-3-1-0300 | MS-2431-3-2-0300 | MS-2431-3-3-0300 |
| Cost | $2.03 | $2.03 | $2.03 |
| Output type | SPST-NO | SPST-NO | SPST-NO |
| Must Operate | 10 ~ 15 AT | 15 ~ 20 AT | 20 ~ 25 AT |

Table 19 Summary of Reed Switch Options

After the initial round of Reed switches were purchased and tested, it became clear that even the most sensitive units still needed to be held very close to the magnets that were used for testing. There was also a change in design that meant we needed to obtain Reed switches that were normally closed and would pull down the microcontroller GPIO pin to ground when the presence of a magnet was absent. Therefore, we ordered one final set of Reed switches from Digikey by Littelfuse Inc (product number: 59025-4-T-02-A), which served as the safety mechanism for our final design.

### Magnet Selection

Investigating the various magnets available yielded a handful of promising options for implementation in this project and the synopsis of some of the best choices are listed below in Table 20. The factors that weighed most heavily in the selection process were size and price (including the minimum quantity that can be ordered). Chances were that we would not have needed to order a minimum of five of any of these magnets in the prototyping process, so all of the options below were selected in part due to the minimum order quantity being one. Since our plan was to affix this magnet near the top of the outside of the water bottle, we wanted to have the magnet be as small as possible so that it would not impede the user from handling the bottle and risk being rubbed off from regular handling. The smallest option was clearly the SMCO5, but the price was over four times that of the 9016 which was the same shape. However, $1.45 was by no means going to be a significant addition to project expenses and we definitely wished to ensure that this magnet would sit as flush as possible to the side of the cylindrical water bottle that will be used for this project. Therefore, it seemed that the SMCO5 1.9X3MM would be the best choice due to its size being the smallest.

Summary of Magnet Options

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | SMCO5 1.9X3MM | 9016 | 8028 | NDFEB 10X5X1.9MM |
| Cost | $1.45 | $0.29 | $0.98 | $1.16 |
| Shape | Cylinder | Cylinder | Cube | Rect. Prism |
| Dimensions | 0.074"D x 0.118"H | 0.079"D x 0.315"H | 0.25" SQ x 0.25"H | 0.394" L x 0.20" W x 0.070" H |
| Magnetization | Axial | Axial | Axial | Axial |

Table 20: Summary of Magnet Options

## Embedded Hardware

### Controller Selection

The controller for the product was always intended to be one of the following: FPGA, DSP, ASIC, or a microcontroller. The controller technologies were investigated and compared, to determine which option is a suitable choice for the project.

#### FPGA

FPGAs are a very capable choice, they typically have a faster release to the market, are relatively affordable, and they allow for flexibility and re-programmability when writing code. But they also require a significant amount of software/coding overhead in HDL, and they are not the best option when it comes to power consumption. Both electrical engineers in this project did have some preliminary experience in the Verilog coding language; however, the programming process could have been made much simpler if it were to be performed in another language that was more abstracted beyond logic gates [46][47]. This project required a sophisticated level of firmware to handle wireless communication, interface with sensors, and report quality data to the user. Because of the aforementioned reasons, an FPGA was not a suitable choice for this embedded system.

#### DSP

DSP chips excel in both real-time processing and quick arithmetic computation. They also typically have a simpler coding process, and are easily reconfigurable since the data is digital, instead of analog. Since this project was initially intended to report water quality based on images represented as parallel output bits, a DSP seemed like a suitable choice for this application. However, DSP chips, due to their high-speed processing, have a higher power consumption compared to a microcontroller. They also require additional filtering and IC’s such as an ADC and DAC. Lastly, DSP chips are typically costly, and the electrical engineers on this project did not have a lot of experience on DSP programming. For these reasons, a DSP was not a suitable choice [46][48].

#### ASIC

Since ASIC’s are mainly comprised of fixed logic gates and digital circuitry, they are very affordable. They also are highly optimized in terms of power consumption and clock cycles. However, since they are designed for one purpose only, they are not re-configurable. Due to their specific functionality, they also typically have a high development overhead [46][47]. For this project, an ASIC would not have been a good choice because the device needed to be able to interface with sensors, process data, and wirelessly communicate via Bluetooth. All these concepts were too individualistic and therefore an ASIC did not currently exist with all of the necessary capability for this project. That meant that an ASIC would have needed to be independently designed for the project. Although the cost per unit would have gone down after development, creating an ASIC from scratch was not feasible. Therefore, an ASIC (during early/prototype phases) was not a suitable choice for this project.

#### SOC

SOC’s are similar to a microcontroller, but they have more specific peripherals. A Bluetooth SOC would have been a great choice for this project. However, it also needed to have enough GPIO pins to support the project, and enough I2C/UART pins to communicate with all of the sensors. An example of an SOC that could have been implemented in this project is Silicon Labs’ EFR32BG22, or the Nordic Semiconductor nRF52840. Both of these SOC’s incorporate Bluetooth 5 and would have therefore been useful for the project. They also both appear to have I2C, UART, and GPIO pins. As long as the SOC’s had enough of the necessary connections, they would have certainly been implementable. However, the main constraints with using this technology were product availability and software overhead. Most of the Bluetooth SOC’s at the time were out of stock (or very limited stock) at major distributors such as Mouser and DigiKey. Using an SOC would have also required becoming familiar with a new architecture and programming interface, which could have been very time consuming. For these reasons, an SOC also was not a suitable choice for this project.

#### Microprocessor (SBC)

One specific brand of Single Board Computer (SBC) is the Raspberry Pi series that is capable of running a fully-fledged operating system known as Raspbian [49]. The product known as “Raspberry Pi Zero W” is described by vendors as a smaller sized version of the Raspberry Pi, but it also includes built in Wi-Fi and Bluetooth capability [49]. The price for these units was $10, but there were additional accessories that would have also been necessary in order to run the device. The Raspberry Pi would have been a good choice for the original spectrometer idea that involved video signal processing.

This controller technology is a microprocessor, which is similar to a microcontroller but has a different organization of device peripherals. A microcontroller has all processing, memory, and I/O in one chip. Whereas a microprocessor consists of a CPU and utilizes a bus for communicating with other peripherals. This allows for designs to be more flexible and incorporate different levels of memory access (depending on speed and capacity). This also would have been a worthwhile addition, if water quality data processing required rapid access to a lot of memory.

Using a product like this would have also provided a lot of capability in terms of processing power and built-in wireless connectivity, but there were still a few issues that needed to be considered. One such problem was that having an entire computer installed within the project device would have undoubtedly drawn more power than a microcontroller with less advanced features. The addition of every peripheral that would have needed to be plugged into an SBC device such as this would also have added to the power requirements of the entire system. This then could have yielded a shorter battery life in the final product. Also, using this technology would have involved a large amount of software overhead for the team. Raspbian is based on Linux OS therefore a lot of research into how to program the device and interface with peripherals would have been necessary. Another constraint that needed to be respected was the size of each component being included in the final product. The width and thickness of the Raspberry Pi Zero W are both fairly miniscule, but the total length of the board is 2.6 inches [50]. Units that had already been investigated for the power system were already well over two inches in length, but it is worth mentioning that the project was supposed to fit in a casing that would serve as a water bottle cap. Therefore, every addition of large circuitry needed to at least carry it weight in functionality.

#### Microcontroller

Microcontrollers are relatively affordable and easy to re-configure. They are a great controller to implement for GPIO functionality, which was favorable for the sensors necessary in this project. They typically support wired communication protocols such as UART, I2C, and SPI. They also often include low power modes which drastically improve the overall battery lifetime. Typically, the architecture of the chip can optimize the clock cycles, which helps to improve the overall efficiency of the device [46]. They also have memory storage, which would have been helpful for possible digital processing necessary for predicting water quality. Microcontrollers are also more cost effective than other aforementioned options. But they do have limitations on the tasks that they can perform [47]. However, this project appeared to have a reasonable scope for implementing a microcontroller. In addition to this, the electrical engineers had the most experience in programming a microcontroller and implementing it into various embedded systems.

Prior to senior design, the electrical engineers on this project had previous experience on working with TI microcontrollers. Both the MSP430FR6989 as well as the MSP430G2553 were utilized in previous embedded programming classes. Due to this familiarity, it was beneficial (as far as time constraint) to implement a microcontroller from the TI family into the project. Other microcontrollers would have required a significant amount of time to understand the user manual and become familiar with the device architecture and software interface. At the beginning of Senior Design, our team also had access to MSP430 development kits. So, selecting an MSP430 microcontroller offered the potential to decrease the project budget.

### Controller Selection Summary

Using the results above, each controller type was compared across the following parameters: Estimated cost, software overhead, market availability, and team familiarity. Each column after estimated cost uses a scale from 1-6. In which a 1 represented the most desirable option, and 6 represented the least desirable option. Table 21 clearly demonstrates that a microcontroller was the most suitable choice for the project. Therefore, more technological investigation on types of microcontrollers will be covered in latter sections of this document.

Controller Selection Summary Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Controller Types** | **Estimated Cost** | **Software Overhead** | **Market Availability** | **Team Familiarity** |
| FPGA | $4 to $20 | 2 | 2 | 2 |
| DSP | $2 to $60 | 3 | 4 | 3 |
| ASIC | > $1000 | 6 | 6 | 5 |
| SOC | $5 to $10 | 5 | 5 | 6 |
| SBC | $5 to $10 | 4 | 3 | 4 |
| Microcontroller | $2 to $10 | 1 | 1 | 1 |

Table 21: Controller Selection Summary

### Spectrometer V1 Project Implementation

In order to have used the V1 Spectrometer for water quality analysis, the embedded device would have needed to be able to handle and composite baseband video signals. Using a CMOS camera, such as the Turbo EOS2 V2, would have required more than a typical microcontroller. The signal (carrying color video and sound) would have been more complex than typical GPIO pins on a microcontroller could support or handle. An additional IC (at the least) would have been necessary for project implementation. For example, the TI TVP5150AM1 would have been able to convert the composite video signal into standard ITU-R BT.656 output (or YCbCr) which then could have theoretically been passed to a microcontroller, vectorized, and processed to determine water quality data. Using such an IC would have then introduced even more software overhead and product cost. Another option was to use a DSP/chip that was compatible with this baseband signal; however, again this would have introduced a large amount of software overhead and it was difficult to find a chip that still had enough peripherals for every other project component. For these reasons, spectrometer V2, which focused on a photodiode and a stepper or servo motor, appeared to be much more manageable than this design.

### Optical Diode PWM Potential Implementation

It was likely that the sanitization UVC LEDs would need to be constantly on, but we initially thought it might make more sense to occasionally observe water quality (instead of at a constant rate). Also, in order to optimize device power consumption and battery life, the optical diodes (from an electrical standpoint) were considered to be pulsed on and off when operating. Diodes that are constantly on create constant current draw which will then decrease the overall battery lifetime. However, this is a feasible sacrifice to make in order to ensure the effectiveness of the UVC sanitization. Nonetheless, we still thought that it would be proactively wise to account for this possible method of operation. Therefore, the selected microcontroller was desired to be able to generate multiple PWM signals. This was mainly because of a possible PWM signal for the stepper motor, as well as the ability to have more signals for the optical diodes.

### Microcontroller Comparisons

#### Desired Microcontroller Feature Set

Prior to investigating the available products on the market today, it was important to first come up with a rough estimate of the desired features that the microcontroller should have. In addition to the features below, it was useful to select a microcontroller (and programming language) that the team was already familiar with. Due to the electrical engineering project focus, having a smaller software overhead helped to streamline the product development process. As a sidenote, most of the packages/variants of each microcontroller considered were selected since they were the most readily available (in comparison to other packages/versions) at the time. Below in Table 22 is the feature set for this project’s ideal microcontroller is outlined in detail.

Microcontroller Feature Overview

|  |  |
| --- | --- |
| Feature | Desired Value/Quality |
| Voltage supply | 2-5 V from on-board regulators |
| Low-Power Draw | < 10 µA |
| Cost | < $10 |
| On-board Wired Comms Protocols | UART, I2C (possibly SPI) |
| Current-Driving Pins | Potentially drive Two UV-C LEDs and green laser directly |
| Low-level Wake-up (LLWU) Pins | Approximately 8 in total (to account for various inputs to the microcontroller). |
| GPIO Pins | At least 20 pins (including LLWU) |
| ADC resolution | 10-12 bit for millivolt precision sensing |
| Analog Input Pins | Water quality sensor (1 pin required) |
| PWM Capability | < 100 µs, at least 2 pins (Servo, and green laser) |
| Interrupts | Operational Safety, and Process Hierarchy |
| Memory | Around 50 kB (Overestimate of water quality data/analysis) |

Table 22: Microcontroller Feature Overview

#### ATmega328-PU

The ATmega328P was easily the most widely available microcontroller (out of the options considered) in the market at the time. It is a very popular microcontroller that is most notably used in the Arduino embedded development kits. It has a supply voltage ranging from 1.8 V to 5.5 V, with six different sleep modes. There is a power-down mode that has a constant current draw of 0.1 microamps when the microcontroller is being operated at 1.8V. The clock can run up to 20 MHz (with 4.5 V supplied). This microcontroller also supports SPI, I2C, and USART; the latter being a similar serial connection to UART with slightly different framing protocols. It has 32KB of program memory, 1 KB of EEPROM, and 2KB of internal SRAM. As far as timing capability, it has two 8-bit timers, and one 16-bit timer. All of these can be run off an internal oscillator. This microcontroller also has 23 I/O lines, six PWM channels, an 8-channel 10-bit ADC, and a built-in comparator. It is also worth mentioning that the data sheet specifies interrupt and wake-up features on pin changes. As far as packaging, it comes in a 28-lead/pin package.

This microcontroller certainly had a lot of desirable attributes for this project. Some of the most important being the market availability as well as the ability to code/flash through the Arduino IDE (with a bootloader). Potentially being able to write code in a simpler IDE such as Arduino and then flash the microcontroller would have also helped to mitigate software overhead for the project. Although, the project members were less familiar with this microcontroller, utilizing the Arduino IDE would still have had a smaller software overhead than trying to learn a brand-new IDE from scratch. This option also had a significant amount of independent PWM signals which could have come in handy if the project required additional PWM signals that were not directly related to the stepper motor. For the scope of this project, it appeared that this option had more than enough timing capability--as long as the microcontroller’s internal oscillator was reliable enough. It also was an added benefit that this option came with a small amount of EEPROM; this would have yielded an instant erasure method (done through the hardware itself). Another great feature of this microcontroller was the extensive amount of power modes. This would have provided more flexibility depending on which peripherals and device internals needed to operate at various times.

It was very likely that the Arduino IDE would have oversimplified device operation. This would have meant that programming with the Arduino IDE would not have as much capability as programming in another environment more closely related to Atmel. Using a new environment such as Atmel Studio would have then involved a larger software overhead than the TI microcontroller options. Extensive time would have had to have been spent understanding the ATmega328P user guide and how to interface with all device internals and registers. Since the project was already intended to have a large amount of software overhead with Bluetooth and app integration; it was preferable to not have to learn an entirely new embedded system. Additionally, this microcontroller had a smaller ADC resolution in comparison to some latter alternatives. It also had a smaller number of I/O pins which could have led to issues during pin configuration later on in the design process. For these reasons, this microcontroller was not the best choice for this project.

#### MSP430G2553IN20

The TI MSP430G2553 was a microcontroller that the team had previously worked with in prerequisite classes. This microcontroller has a supply voltage from 1.8 V to 3.6 V, and an ultra-low power mode of 0.5 microamps (with five additional power saving modes). It also supports serial communication such as I2C, UART, and SPI. It has a built-in ADC with 10bit conversion, a comparator, as well as 16 I/O pins. As far as timing capability, this microcontroller has an internal LF oscillator as well as an internal 32-kHz crystal. It is capable of frequencies up to 16 MHz, and has two timer modules, each with three capture and compare registers. Each CCR of the timer modules appeared to have its own PWM signal. As far as the available packages, it comes in 20-pin, 28-pin, and 32-pin SMT packages. However, the 20-pin IN20 package appeared to currently have the largest quantity in stock.

This would have been the cheapest option considered; since economic constraint was very important for this project, component cost did heavily weigh-in to microcontroller selection. Additionally, team members already possessed the development board for this microcontroller, which would have helped reduce development costs on the project. It also was a very widely available microcontroller in the market at the time. Again, due to microcontroller scarcity, that was very important to consider when designing the prototype device. The electrical engineers for this project were most experienced in the TI family of embedded programming. So, the MSP430G2553 would have had some of the smallest software/development overhead. It also had a smaller device footprint and would have allowed for easier component incorporation on the PCB.

If this microcontroller was selected, there certainly would have been a few flaws. It was possible that the microcontroller would have ended up having a scarce amount of GPIO pins, depending on the pin variation/configuration. So, it was wiser to select a microcontroller that had more available pins and configuration flexibility. This option also potentially did not have enough flexibility from a timing standpoint since it only had two timer modules--some of the various timing requirements for the project could have been limited. Also, this option had a slightly smaller bit-resolution ADC which would have yielded less resolution for the water quality sensing. That was a critical flaw when choosing this option since that particular sub-system was thought to be imperative and required high precision (in near microvolts). This microcontroller choice also had a significantly smaller memory size which also could have impacted the ability to track water quality data. Even though 16 KB may have been enough for the project, it certainly was wiser to have a surplus of memory for development overhead instead of constantly maxing out memory capacity. These were the main flaws with selecting this particular microcontroller.

Even though this microcontroller appeared to have some flaws, it was still widely available in the marketplace and appeared to be potentially capable enough for the project. So, this microcontroller was highly considered for this project, and it was therefore important to consider the proposed pin configuration. The table below represents a proposed pin configuration for the 20-pin version of the microcontroller that contains: UART Tx and Rx, 8 Interrupt GPIOs, 1 Analog I/P, and 2 PWM outputs. Red font color in the table below represents the selected configuration for each pin (mainly related to peripherals). There were other connections (required for flashing the microcontroller) that were not selected since they were similar across every microcontroller with proposed pin configurations.

Proposed 20-Pin G2553 Pin Configuration

|  |  |
| --- | --- |
| 20-pin G2553IN20 | |
| Pin # | Configurations |
| 1 | DVCC |
| 2 | P1.0/TA0CLK/ACLK/A0/CA0 |
| 3 | P1.1/TA0.0/UCA0RXD/UCA0SOMI/A1/CA1 |
| 4 | P1.2/TA0.1/UCA0TXD/UCA0SIMO/A2/CA2 |
| 5 | P1.3/ADC10CLK/CAOUT/VREF-/VEREF-/A3/CA3 5 |
| 6 | P1.4/SMCLK/UCB0STE/UCA0CLK/VREF+/VEREF+/A4/CA4/TCK |
| 7 | P1.5/TA0.0/UCB0CLK/UCA0STE/A5/CA5/TMS |
| 8 | P2.0/TA1.0 |
| 9 | P2.1/TA1.1 |
| 10 | P2.2/TA1.1 10 11 P2.3/TA1.0 |
| 11 | P2.3/TA1.0 |
| 12 | P2.4/TA1.2 |
| 13 | P2.5/TA1.2 |
| 14 | P1.6/TA0.1/UCB0SOMI/UCB0SCL/A6/CA6/TDI/TCLK14 |
| 15 | P1.7/CAOUT/UCB0SIMO/UCB0SDA/A7/CA7/TDO/TDI |
| 16 | RST/NMI/SBWTDIO |
| 17 | TEST/SBWTCK |
| 18 | XOUT/P2.7 |
| 19 | XIN/P2.6/TA0.1 |
| 20 | DVSS |

Table 23: Proposed G2553 20-Pin Configuration

Using the proposed pin configuration above, this would have only yielded 3 remaining GPIO pins (with interrupt capability). This left very miniscule room for adjustment in the future. And, it also certainly would not have worked for the project. If a board revision required significant overhaul of any subsystem needed to be performed, it would have required assigning a lot more than three additional GPIO pins. Therefore, even though this microcontroller was cheaper and had a larger quantity in stock at the time, it was still not the right choice for this project. This was because the configuration was already incredibly limited, and it did not have any room for additional feature sets in the future.

#### MSP430FR6989IPN

Not only was this the most familiar microcontroller considered, but it also ws one of the most capable. It supports a wide supply voltage range from 1.8 V to 3.6 V, as well as optimized ultra-low-power modes. The stand-by power mode with the low frequency oscillator is 0.4 microamps. It also has a real-time-clock low power mode with a typical current draw of 0.35 microamps. The device comes with 128 KB of memory; it is worthwhile to note that this memory is non-volatile. As far as timing, this microcontroller comes with five 16-bit timers with up to seven capture/compare registers each. It appears that each capture/compare register is capable of having its own PWM signal. There is also a 12-bit ADC with 16 input channels, and a 16-channel analog comparator. Ports 1,2,3, and 4 are all capable of waking up the microcontroller from low power mode. A port is typically comprised of 8 GPIO pins; therefore, 32 pins in total are capable of waking up the microcontroller (synonymous with Low Level Wake Up terminology). This option also comes with enhanced serial connection which boasts UART with automatic Baud-Rate detection, SPI, and I2C with multiple-Slave addressing. There is also a built-in UART and I2C bootloader. The clock system in this microcontroller also has a lot of flexibility. There is a fixed frequency DCO (with 10 pre-set frequencies), a low power and low frequency internal clock source, and low frequency and high frequency crystals. The MSP430FR6989IPN has either 63 I/O pins. This microcontroller comes in an 80-pin or 100-pin package. However, the 100-pin package was very scare at the time and also the 80-pin package seemed sufficient for the project. The microcontroller had more than enough features and pin configurations, and it would have created a smaller footprint than the 100-pin version would on the printed circuit board.

The largest advantage that this microcontroller offered was a lower software overhead, since both project electrical engineers took an entire embedded programming class that utilized this particular microcontroller. This then indicated that the majority of software development would have involved incorporating Bluetooth technology and general control logic into the project. This option also offered the largest amount of timing modules; it would have certainly been helpful since multiple unique PWM signals might have needed driven from the microcontroller at different times throughout typical device operation. It is always helpful for the microcontroller to have a wide supply voltage range. Depending on the battery and regulators utilized, being able to run the microcontroller at a lower supply voltage (with a lower clock frequency), was thought to possibly improve battery life. This microcontroller had more than enough memory for the scope of this project. Selecting this option would have also provided the most ADC resolution which was certainly favorable for determining water quality and/or the presence of biological contaminates in the water. Having 32 different pins with capability to wake the microcontroller was also nice; this was thought to allow for the microcontroller to sleep until the button is pressed to start sanitization or analysis.

The largest downside to using this microcontroller was that it was too capable for the scope of the project. This project most likely would not have required too many I/O features. However, this was ultimately a good problem to have since it would have allowed for the largest amount of flexibility when deciding upon pin configuration. In other words, there would not have been a shortage of capability when it came to PWM, external interrupts, serial communication, etc. This microcontroller was essentially guaranteed to work for the project without any sort of functional bottleneck (in regard to pin configuration). Additionally, later on in the project the scope was more well-defined, and more features were to be added. So, in selecting this microcontroller, there would have been more assurance that the project would still be implementable and manageable in the near future. Lastly, this microcontroller was the most expensive option. But it also had a lot of value and timesaving to offer so it appeared to be more than worth it. Also, the high price appeared to be due to the scarcity of this particular model in the market at the time. So, it seemed wise to look into other variants similar to the 6989 microcontroller that were more available. Shown below in Table 24 are the preliminary microcontroller comparisons that were performed on the aforementioned candidates, with Team Familiarity and Software Overhead scores, where higher numbers are better.

Microcontroller Comparisons 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Controller** | **ATmega328-PU** | **MSP430G2553IN20** | **MSP430FR6989IPN** |
| **Pins** | 28 | 20 | 80 |
| **Active Mode Current** | 200 µA/MHz @ 1.8V | 230 µA/MHz @ 2.2V | 100 µA/MHz @ 3V |
| **Standby Mode Current** | 0.1 µA @ 1.8V | 0.5 µA @ 2.2V (LPM3) | 0.4 µA @ 3V (LPM3) |
| **Price** | $2.63 | $3.38 | $9.43 |
| **Availability** | 46,400 (DigiKey) | 7400 (TI) | < 300 (DigiKey) |
| **Software Overhead** | 3 | 2 | 1 |
| **Team Familiarity** | 3 | 2 | 1 |
| **Supply Voltage** | 1.8 V to 5.5 V | 1.8 V to 3.6 V | 1.8 V to 3.6V |
| **Wired Comm. Protocols** | I2C, USART, SPI | I2C, UART, SPI | I2C, UART, SPI |
| **Current Driving** | 20 mA (Vcc=3V, Vo=2V) | 25mA (Vcc=3V, Vo=2V) | 25 mA (Vcc=3V, V0=1.5V) |
| **GPIO Pins/Interrupt** | 23/23 | 24/16 | 63/32 |
| **Analog Input Pins** | 8 | 8 | 16 |
| **PWM Output Signals** | 6 | 6 | > 6 |
| **ADC Resolution** | 10-bit | 10-bit | 12-bit |
| **Timing Modules** | 3 | 2 | 5 |
| **Memory** | 32 KB | 16 KB | 128 KB |
| **Max Clock Speed** | 20 MHz | 16 MHz | 16 MHz |
| **Development Board** | Arduino UNO | Launchpad | Launchpad |

Table 24: Microcontroller Comparisons 1

After assessing the aforementioned microcontrollers, it appeared that the MSP430FR6989 was the best candidate for this project at the time. However, there were other microcontrollers made by TI that seemed similar to the MSP430FR6989. In the next section, two more alternatives were considered for microcontroller implementation in the project.

#### MSP430F5529

This microcontroller was recommended by one of our project supervisors—Dr. Richie. It was initially thought to provide similar capability to the MSP430FR6989 in a smaller and more low-cost package. It uses a low supply voltage range from 1.8V to 3.6V. This microcontroller also has low power mode functionality with a slightly higher sleep current than the FR6989. It comes with a real time clock, crystal, and low-power oscillator (VLO). It has a typical standby mode current (in LPM3) of 1.4 microamps when the device is being operated at 3.0V. The clock speed can go up to 25MHz. The microcontroller has four different timers, each having anywhere from 3 to 7 capture/compare registers (CCRs). As far as serial communication, it supports UART, SPI, I2C, as well as even USB. The ADC has 12-bits of resolution, 14 external channels, and there is also a built-in comparator. It has 63 I/O pins, and it comes with 128 KB of flash and approximately 8 KB of SRAM. This microcontroller also has serial onboard programming, and it does not require external programming voltage. This device only appeared to come in one package, which was a LQFP 80-pin package.

This microcontroller did appear to be a good choice for project implementation. Most of its feature sets were very similar to the MSP430FR6989 microcontroller that was already previously investigated. It appeared that this microcontroller had a higher clock speed with a differential of +5MHz over the 6989 microcontroller. It also appeared to have more than enough timing capability for this project; four timer modules with a plethora of channels would have been very useful for timing modes of operation. It could have driven enough PWM signals for project implementation. The 5529 also had more current driving capability on its output pins. If any optical components or LEDs were going to be driven directly from the microcontroller, this would have certainly been a useful addition for the project. The 5529 also had the same amount of flash memory and ADC resolution as the 6989. Lastly, the 5529 was significantly cheaper than the 6989 microcontroller.

If this microcontroller was utilized, an entirely new development board would have needed to be purchased for each programmer. This would have added a significant amount to project expenses. Not only in monetary value, but also in the amount of time it would have taken to become acclimated with a new launchpad. The 5529 also appeared to have less GPIO pins with interrupts, so there could have been future limitations on device interrupt functionality. It was also worth considering that this microcontroller had a higher sleep current than the 6989, which would have ultimately had an effect on battery life of the finished product. The 5529 was still an 80-pin microcontroller which seemed a bit excessive for the scope of this project. It was more beneficial to find a microcontroller that had fewer than 80 pins. However, the most crucial flaw for project implementation of this microcontroller was product availability. At the time, TI was out of stock of the 5529, and DigiKey had 1000 on order. Therefore, this microcontroller was not a suitable choice for this project.

#### MSP430FR5989IRGC

This microcontroller was in the same family as the 6989 but had slightly less capability. It still has the same wide supply voltage range from 1.8 V to 3.6 V. It also has the same max clock frequency of 16-MHZ. The active mode current draw is 100 microamps per megahertz. The standby current in LPM3 with the VLO is 0.4 microamps. The microcontroller also has 128 KB of flash memory. As far as timing, there is a built-in RTC with calendar and alarm functions; there are also five timers (16-bit) with up to 7 compare/capture registers each. There is a fixed frequency DCO (with 10 different options), a VLO, and low frequency and high frequency crystals. It also supports 16-bit and 32-bit CRC checks. There is a 16-channel analog comparator, a 12-bit ADC, and 16 external analog input channels. If an LCD was ever to be implemented in the project, it was assuring to know that it did have its own built-in LCD Driver that supported 320 segments, as well as contrast control. The microcontroller has 48 I/O pins, of which at least 32 pins have interrupt capability. Again, these interruptible GPIO’s can be found on ports 1-4. There also still programmable pullup and pulldown resistors on all GPIO ports. As far as serial communication, SPI, I2C, and UART are still supported. The same development tools for the rest of the TI MSP430 family, could still be utilized if this microcontroller were to be selected. Both packages of the 5989 are 64-pins, however this one is a VQFN package with the smallest body size out of all 5989 and 6989 microcontrollers.

In comparison to the MSP430FR6989, the MSP430FR5989 did offer a few additional project advantages. One of them being a smaller package and pin-size which could have attributed toward a smaller footprint on the PCB. Since the printed circuit board had to fit inside of a water bottle cap, it was important to conserve as much space as possible. This microcontroller was also significantly cheaper than the 6989 -- by a factor of three. The software overhead for this microcontroller was also comparable to the overhead that the 6989 would have; this was because they were from the MSP430 family, and they could be programmed to use the same ports. The other main advantages to selecting this microcontroller were pricing and availability. At the time, it was very difficult to find the MSP430FR6989. It would have costed upwards of $10 per unit from a very limited supply across online electrical component distributors. However, the MSP430FR5989 was widely available at TI and for a much more reasonable price.

One of the only downsides to selecting this microcontroller was that it did not have as many I/O pins as the 6989. However, it appeared that the 5989 had more than enough GPIO capability for the purposes of this project. Also, the 5989 did not have as much current driving capability as the 5529. But the 5529 was very scarce in the market, and most likely optical components were also likely to be enabled from a MOSFET switching circuit connected to the microcontroller.

Microcontroller Comparisons 2

|  |  |  |
| --- | --- | --- |
| **Controller** | **MSP430F5529** | **MSP430FR5989IRGC** |
| **Pins** | 80 | 64 |
| **Active Mode Current** | 290 µA/MHz @ 3.0V | 100 µA/MHz @ 3V |
| **Standby Mode Current** | 2.1 µA at 3.0V (LPM3) | 0.4 µA @ 3V (LPM3) |
| **Price** | $5.50 | $3.09 |
| **Availability** | 1000 on order (DigiKey), 0 (TI) | 35,000 (TI) |
| **Software Overhead** | 2 | 1 |
| **Team Familiarity** | 2 | 1 |
| **Supply Voltage** | 1.8 V to 3.6 V | 1.8V to 3.6V |
| **Wired Comm. Protocols** | I2C, UART, SPI | I2C, UART, SPI |
| **Current Driving** | 40 mA (Vcc=3V, Vo=2V) | 25 mA (Vcc=3V, V0=1.5V) |
| **GPIO Pins/Interrupt** | 63/16 | 48/32 |
| **Analog Input Pins** | 14 | 16 |
| **PWM Output Signals** | > 6 | > 6 |
| **ADC Resolution** | 12-bit | 12-bit |
| **Timing Modules** | 4 | 5 |
| **Memory** | 128 KB | 128 KB |
| **Max Clock Speed** | 25 MHZ | 16 MHZ |
| **Development Board** | Launchpad | Launchpad |

Table 25: Microcontroller Comparisons 2

#### Microcontroller Selection Summary

It was determined that the MSP430FR5989 was the overall best choice for the project. It had the capability of the MSP430FR6989, with a slightly smaller package size and fewer pins. However, for this project there appeared to be more than enough functionality for this microcontroller to be utilized. It also had more than enough timing capability, interrupt GPIO functionality, and low-power modes. The Atmega328P was widely available, but it would have required learning a new development platform, and family of microcontrollers. Ultimately, focus shifted toward the TI family of microcontrollers. This was because the electrical engineers on this team were already familiar with rudimentary TI embedded microcontroller applications. The MSP430G2553, was not capable enough and did not allow for design flexibility in the future. After creating a proposed pinout, it was concluded that this option would be too limited for the scope of this project. It also had a much smaller number of timers, and a larger software overhead than the 5989 and 6989 microcontrollers would have had in this project. The MPS430FR6989 was scarce in the market, and it would have taken up a larger footprint on the printed circuit board. It also appeared to be too capable for the scope of this project (in a wasteful rather than practical way). Not to mention it was incredibly expensive in comparison to other project options. The MSP430F5529 would also have worked, but it was unavailable in the current market, and would have required purchasing new development boards. So, it made the most sense to go with the MSP430FR5989. This chip was similar to the 6989, which team members were already the most familiar with. It provided flexibility and capability for overall design and project development. It was more affordable, and it was available in large quantities at the time. Lastly, project members already had the MSP430FR6989 Launchpad development board. This, in tandem with regulated code versions and repository management, would help to streamline hardware and software development. In further sections, the embedded prototyping, pinout, and hardware design for this particular microcontroller will be discussed in greater detail.

### Microcontroller Development

#### Documentation

It can become tedious to find certain details during the development process, so it was important to always have an idea of which documentation needed to be further analyzed in order to accomplish a certain task. TI provides several different documents that were all necessary to review at some point. The first is the datasheet for the microcontroller. It contains specific electrical characteristics, as well as device pinouts, pin configurations, and packaging/manufacturing information. The next piece of documentation is the family users guide. This lays out how to program the microcontroller, how to write the code, and how to interface with microcontroller peripherals. The last piece of documentation is the Launchpad user guide. This guide contains specific details about the development kit itself, and how all of the various components are connected and arranged on the development board.

#### Development Board

For this project’s breadboard prototype, the MSP-EXP430FR6989 Development Kit from TI was utilized. This board comes with the 100-pin (IPZ package) MSP430FR6989, which is effectively a larger version of the final microcontroller that was implemented in this project. Fortunately, TI stated that this development kit was compatible for development with MSP430FR5989 microcontroller, which was the microcontroller selected for this project. Both of the aforementioned microcontrollers use the same MSP430 platform and header file, therefore the code was easily interchangeable between protype and final board layout. Also, it worked out by circumstance that each project programmer already has this development kit. This ultimately helped reduce overall project development costs.

This development kit allows for simple flashing of the microcontroller via USB connection to the PC. It also has 40 external headers that can be utilized in a breadboard prototype. When selecting the desired protype pin configuration, it was important to select ports that have external jumpers, so that the breadboard design could be created. In addition to this, the board also had buttons and LEDs that were used to simulate project components. This decreased the number of components that had to be placed on the breadboard, and it also helped simplify the protype design. The board was also capable of providing 3.3V and 5V DC rails for any electrical/optical prototype components. This also eliminated the need for other external power sources.

### Bluetooth Technology Comparisons

#### Bluetooth 2.1

Bluetooth 2.1 is an earlier version, but still appears to be very popular in the industry today. It is a low-cost option, with enhanced data rates up to 3 Mbps. Version 2.1 also improved device pairing procedure (and security); this decision drastically helped to improve user operation. This particular version has a range of 33 meters [13]. Due to the larger power consumption of this version of Bluetooth, it was not a suitable choice for the project.

#### BLE

Also known as Bluetooth 4, this version of Bluetooth is the first version to drastically improve device power consumption. BLE has a data rate of 1Mbps. It also approximately has the same range as Bluetooth Classic (depending on throughput). This version of Bluetooth would implement lower power consumption and therefore improve the overall battery life of the device. Although this version has a lower throughput, it was considered to be enough bandwidth to push through all of the necessary data for this project. Additionally, a pairing range of 10 feet was more than achievable with this version of Bluetooth. Since this version was intended for devices that only need to send periodic data, this also indicated that this version would be a great choice for the project [52].

#### Bluetooth 5

This was the newest version of Bluetooth available at the time. It still has a focus on low power features, but also increases the data rate and device range. This version offers speeds up to 2Mbps, but also has a high-range option of 125 kbps. If this version were to be used, most likely the high-speed option would have been selected. Since, the range for this product was intended for proximity. This version was developed for an industry that is heavily based in IoT, which has some relevancy with the goals of this project [52]. Bluetooth 5, due to its new arrival, was the most expensive option, and appeared to be more than capable for the project. Therefore, it appeared to be unnecessary to go with this version, since BLE already seemed to be capable enough for this project.

#### Bluetooth Module 2021 Availability

Previously it was stated that a Bluetooth 4.0 Module would be more than sufficient for this project. However, at the time Bluetooth modules (in general) were somewhat difficult to acquire in the market. Since, Bluetooth 5.0 also had the feature set as Bluetooth 4.0, either version would have worked just fine on this project. In a market suffering from chip shortage, product availability remained one of the most important deciding factors for the PCB prototype design. Therefore, a mixture of 4.0 and 5.0 Bluetooth modules were further investigated in regard to design implementation.

### Bluetooth Module Investigation

Upon beginning the technology investigation for Bluetooth Low Energy (BLE) modules, it was originally hoped that purchasing a prefabricated unit would allow for relatively simple integration and communication with the microcontroller that had been chosen for this project. It quickly became apparent that one of the main challenges in implementing Bluetooth capabilities in this project was going to be programming the firmware that runs on the selected module. As is has been stated many times previously, none of the members of this project team have had any experience in employing Bluetooth communications in any previous assignments or personal ventures. Therefore, it was of utmost importance that any module chosen for this project would be as simple and straightforward to use as possible so as to ensure that our desired features could be fleshed out within the given time constraints of this endeavor.

#### CYBLE-013025-00

When this device was first discovered, it seemed to be a standard BLE module that could interface with our microcontroller by either using I2C, SPI, or UART. This model of BLE module utilizes the Bluetooth 4.1 protocol, so we would not be getting the newest and most secure version for communicating with smartphone devices. The module is a Surface Mount Device (SMD) that uses the ARM Cortex M3 as the core and is manufactured by Cypress Semiconductor. Also, the CYBLE unit is listed as being shielded which would protect against RF interference. The device is listed as moisture sensitive, but it was originally unclear how many SMD BLE modules were available that did not have this limitation.

Upon reading into the datasheet, it could be seen that there were two ways that a user could begin to get working firmware onto the device: by either using the Wireless Connectivity for Embedded Devices Smart Software Development Kit to design the firmware needed for project integration or to take advantage of the EZ-Serial™ Firmware Platform (8). The EZ-Serial™ Firmware Platform is described on Cypress’ website to allow the use of their compatible products without the need for prior knowledge of Bluetooth stack or programming [54]. Their website claimed that there is no need for an IDE to have “Out-of-the-box support for CYSPP mode with no special configuration”[54]; it was not mentioned anywhere on the webpage what “CYSPP mode” is but the user guide for this firmware defines this as “Cypress Serial Port Profile” mode (11). Therefore, at first the claim of working “out of the box” seemed dubious, but it was encouraging to find that this firmware is already programmed onto the module in question (8). It is also worth mentioning that the documentation on this firmware mentions that it is possible to communicate with the module via UART or GPIO control [54]. Considering the fact that this module has dedicated GPIO pins for SPI and I2C communication (8), it seemed to be a reasonable assumption that it would be possible to interact with the pre-installed firmware by using these two communication protocols.

#### RN4871 BLE Module

This unit made by the Microchip company was another very tempting option for this project. On the product website, it boasts an ASCII command interface for simple interaction with microcontrollers along with a mobile app for interacting with the device. This module is certified for Bluetooth version 5, which is listed as being able to provide more throughput and more secure connections when compared to Bluetooth 4.1. It was not clear at first if it would be necessary to have a faster connection to the Android app in order to achieve the desired performance but since the analysis for water quality was completed by the microcontroller, it seemed that the transmission of the final results would not be the limiting factor in the successful project execution. Reading though the datasheet of this device, a recommendation was found for the proper placement of the Bluetooth transmitter on the PCB that was created (9). Regardless of which module was chosen to satisfy the wireless communication needs for this design, it became clear that proper placement of the transmitter would play an important role in the creation of the board in order to optimize the performance of the device. It was difficult to determine the extent to which premade app called Microchip Bluetooth Data would meet the needs of our project without testing but according to the Google Play store description, this app has the ability to “Scan and connect LE device. Transfer text typed in the app to peripheral device. Transfer text file data, send and receive across the device and phone” [55].

#### EYSPBNZUA

This BLE module was over twice as expensive as the Cypress option and in much shorter supply, however it had some very enticing improvements that were worth considering. Firstly, the average current required during data transmission is almost half that of the CYBLE unit while also having a 26.6% lower current draw during low-power-mode. The combination of these two upgrades would certainly have meant getting a noticeably longer battery life from the final product.

#### Bluetooth Module Comparison/Selection

When examining the three options that were investigated, the biggest factors in the final choice of module had been gathered in Table 26 below for convenience. It was highly desired to utilize the RN487x series of modules due to the pre-made smartphone app that could have potentially eliminated the need for app design and coding entirely. However, it was found that the availability of these chips was just not going to meet the time constraints of this project.

Therefore, the final two options needed to be evaluated based off of the needs for this project. It would have been much more energy efficient to go with the EYSPBNZUA module, but there were major concerns about whether this energy demand decrease would be worth the price being over twice that of the competitor. Considering the fact that the module only transmits for very limited time frames after the water sensor has been activated, it seemed that the drastic difference in transmitting supply current would not lead to much of a difference in terms of battery life for the device as a whole. It was also discussed that the exclusive use of Bluetooth Low Energy mode for this project would mean that the Bluetooth 5 functionality would not even be utilized. The Cypress module was also much more available in terms of stock, and this was a reassuring factor for the ability to actually obtain the module. It would have been very damaging for the time constraints of this project to dedicate a significant amount of effort into investigating how to implement a Bluetooth module for it to end up being bought out from underneath us and to start from scratch to figure out how the Cypress module works. Finally, it is worth mentioning that the documentation for the EYSPBNZUA module does not mention any firmware or API that is designed for simple and out-of-the-box operation.

In conclusion, it seemed that the CYBLE-013025-00 BLE module would be the best option for the needs of this project. In the following sections, it is discussed how to use the firmware that comes installed in this Cypress module as well as the potential need for an evaluation board for the development phase of this project.

BLE Module Comparisons Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Controller** | ***CYBLE-013025-00*** | **RN4871** | **EYSPBNZUA** |
| **Supply Current Transmitting** | 26 mA | 13 mA | 14.2 mA |
| **Lowest Power Mode Current** | 1.5 µA | 60 µA | 1.1 µA |
| **Price** | $5.94 | $7.38 | $12.70 |
| **Availability** | 881 | 16 | 76 |
| **Shielded** | Yes | Yes | Yes |
| **Moisture Sensitivity Level** | 3 | 2 | 3 |
| **Bluetooth Protocol Version** | 4.1 | 5.0 | 5.2 |
| **MCU Interface** | I2C, UART, SPI | UART, AIO, PIO | I2C, UART, SPI |
| **Earliest Shipping Arrival** | 2nd day, overnight | December 24, 2021 | 2nd day, overnight |

Table 26: Bluetooth Module Comparisons

### CYBLE-013025-EVAL EZ-BLE™ Module Arduino Evaluation Board

The EZ-Serial WICED BLE Firmware Platform User Guide specifically recommends using the appropriate evaluation board for learning and developing with the CYBLE-013025-00 module (11). Being able to connect this evaluation board to Tera Term or any other serial terminal software that is available for a PC greatly expedited the process of team members familiarizing themselves with the Application Program Interface (API) that is available for interacting with the selected Bluetooth module. Since none of the members of this project team had any prior experience in programming for devices to have Bluetooth interconnectivity, it seemed highly prudent for at least one of us to begin learning and preparing code for the microcontroller to interact with the Cypress device. This evaluation board was listed at $50 on Cypress’ website, but it was also mentioned that there was a possibility to obtain samples or reduced rates for students. An email was sent out to a Cypress sales representative for further information, because adding $50 of development cost for just one evaluation board was a very steep inclusion for a list of production costs that was already rising rapidly from the initial estimates. The representatives never got back to us, so we ended up buying the board for evaluation purposes.

### EZ-Serial WICED BLE Firmware Platform

This preloaded firmware was one of the main selling points for the Cypress BLE module that was selected, yet it still took a fair amount of effort to ensure that the default settings were going to provide the necessary functionality for this project while also learning how to properly interact with the device on both the microcontroller end as well as on the intended smartphone app. It was possible to install a custom firmware image on the CYBLE-013025-00 if the standard platform did not meet our needs by using the WICED Smart SDK Integrated Development Environment (IDE) (11), but this was only going to be a last resort measure since it would have taken a considerable amount of programming overhead.

Parsing through the firmware user guide, it can be seen on page 8 that the firmware on this module has some settings initialized that are important to remember for all coding purposes (11), and these settings are shown below (see Table 27).

EZ-Serial WICED BLE Firmware default UART Settings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Baud rate** | **Data bits per frame** | **Parity bit enabled?** | **Number of stop bits** | **UART flow control** |
| **Default Setting** | 115,200 | 8 | No | 1 | Disabled |

Table 27: Default UART settings for Cypress BLE module

Unfortunately, the selected module did not support a text-based output for the API and the units that do support this mode were not available due to the limited stock. This meant that all of the information that gets passed to and from the BLE module via the built-in API would be shown in the serial terminal as a string of hexadecimal values that describe the API event that has occurred. Thankfully, it is mentioned on page 9 of the user guide that there is a Python script that is available for free on Cypress’ website which would convert between text commands and the binary (hexadecimal) messages (11). The particulars about how this Python code works is explored in a later section of the user guide and this report.

There are two main modes that the device can operate in that are selectable via the CYSPP GPIO pin by passing either digital low for Cypress Serial Port Profile (CYSPP) mode or digital high to activate command mode (page 10, 11) . Communicating with the device in command mode is done by sending binary (hexadecimal) data packets with a format that is described on page 11 of the user guide (11). More detail can be explained for command mode if it is necessary, but it seemed that CYSPP mode would be the method through which we will be able to actually send data over a Bluetooth Low Energy connection (page 14, 11). CYSPP mode can be activated in three different ways, but it seemed that it would be easiest to do so by passing a digital low to the CYSPP pin via the microcontroller, as it proved to be.

When operating in CYSPP mode, the Cypress module was only able to operate in a peripheral role (page 14, 11). Again, if this ended up not offering the functionality that was needed, it was possible to create a custom program by using the WICED Smart SDK IDE. When a device that is not made by Cypress is to be used to serve as the central role in the BLE communication with the module, the user guide indicates that it needs to be configured to follow a specified procedure. There are examples given for configuring a device to connect to the module in CYSPP mode in section 3.2 of the user guide (11).

When the Cypress module starts up in CYSPP mode, it begins “advertising” with the preconfigured settings. According to the Bluetooth website, advertising for Bluetooth Low Energy is either done to establish a two-way link between devices or to broadcast data without establishing a connection [56]. The EZ-Serial user guide then says that once a wireless connection has been made between the module and the other device (in this case it will be a smartphone), the smartphone must “subscribe” to either acknowledged or unacknowledged data (page 15, 11). The device is also given an optional choice to enable receiver flow control so that each end can communicate when it is ready to write new data (11). For this project, we wanted to select both acknowledged data as well as RX flow control in order to maximize the reliability of data transfer. Choosing these settings does reduce the potential throughput of data, but the speed at which the app responds was not as important as the integrity of the data when said data indicates the potability of water. After the connection has been established, the EZ-Serial firmware asserts the “CONNECTION” pin in order to tell the microcontroller that it is ready to transmit and receive data (page 15, 11). There are multiple ways listed to close the data pipe between the Cypress module and smartphone, but the option of interest for this project was to maintain the connection until the smartphone disconnects.

Moving on to the operational examples in the user guide, there is a detailed discussion about how to select baud rates, enable flow control, and manage the sleep states of the Cypress module by interacting with the API vie UART in command mode. Most of this information is of interest but will be reserved for discussion during the project design portion of this report concerning any changes that could have been made to the standard settings of the EZ-Serial firmware. It is described on page 27 of the user guide that since the factory default for the BLE module firmware is configured to be in “auto-start” mode, the Cypress device begins automatically advertising to connect to the smartphone as soon as it is powered on (11). The guide also indicates again that the serial data stream is bi-directional, which means that even though the Cypress module can only function as a “GAP Peripheral” with the EZ-Serial firmware, this still allowed the module to take in data over UART from the microcontroller and to send it via the BLE protocol to the smartphone app.

Much of the rest of the user guide for this firmware describes commands that can be sent to the Cypress module API while the device is in command mode. It did seem possible to update data stored in a “local GATT Server attribute” on the BLE module by using the “gatts\_write\_handle” command and then tell the smartphone client that this data has been updated by using the “gatts\_indicate\_handle” command (pg.92-93, 11). However, the module does not automatically connect to a client wirelessly while in command mode, so it seemed that the way to utilize this automatically established data pipe would be to only operate in CYSPP mode. Since we wanted to send values of a fixed data type to the smartphone as indicators of water quality, it seemed to make sense to use “mode 2: fixed” setting for CYSPP transmission packetization. If the module waits until it has received a fixed number of 20 bytes over UART from the microcontroller, then we would know for sure that each data packet being sent to the smartphone app would have that many bytes of useable data to be extracted, interpreted, and displayed. Reading through this user guide has clarified that the selected Bluetooth module would be able to connect to a smartphone and transmit data packets that are sent to it via UART.

### Bluetooth Module Hardware Implementation

The PCB design and pinout connections to the microcontroller both needed to account for the GPIO pins of the Cypress BLE module that we were connecting to. The intention of this project was to keep the communication between Bluetooth module and microcontroller as simplified as possible in order to minimize the programming overhead that needed to go into the Bluetooth communications aspect. However, it was highly prudent to prepare for the more advanced interoperability between these two devices in case a need arose for the more advanced functionality that the Cypress module is capable of. The bare minimum number of connections that needed to be made between microcontroller and the BLE module were only two lines for transmission and receiving. In the design of this project, it was decided that incorporating the “Connection” and “Data Ready” outputs from the BLE module would be sensible for the purpose of writing more robust communication code for the microcontroller. Since we had already selected a microcontroller that had plenty of GPIO pins to spare, it seemed prudent and wise to include connections to some of the BLE module GPIO pins that we did not initially intend to use. In this way, we left ourselves the opportunity to implement some of the more advanced features of the module, if the need did arise. Therefore, the inclusion of connections for the flow control (CTS and RTS), mode selection (CYSPP), low power mode, and low power mode status pins were a precautionary measure that was taken to reduce the impact if the project required redesign further down the line.

### Wired Communication Protocols

Upon investigating the various options for Bluetooth modules available on the market, it became clear that there were multiple choices for how we would pass data from the microcontroller. In order to maximize safety and performance of the water quality sensor, it was important to ensure that we could minimize the risk of data corruption during the transmission process from the microcontroller to the BLE module, and to the smartphone application.

#### UART

A few of the Bluetooth modules that were explored as possibilities for this project had suggested that using the simplified firmware that comes pre-installed on the devices would limit the communication to said device to the Universal Asynchronous Receiver/Transmitter (UART) protocol. UART may be one of the most simplified wired communication protocols, because it is a form of serial transmission that only needs two wires between the host and client [57]. Unfortunately, simplicity can sometimes mean that sacrifices are made in terms of the quality of the data transmission.

The greatest concern that needed to be addressed if the UART protocol was going to be used was ensuring a high degree of integrity of the communication between the microcontroller and the BLE module. The asynchronous aspect of UART transmission meant that the bitstream being sent between host and client were not being synchronized with the frequency of the clock signal; instead, there was a sampling method that the receiving end of each communication used that was actually driven by the clock signal [57]. There are fixed baud rates that have been decided upon that are used by any UART implementation [57], which simplified the process of choosing speeds for the client and server. However, neglecting to ensure that the two devices using UART have been initialized to transmit and receive at the same baud rate would have inevitably lead to sampling errors in which the receiving end was not properly collecting the incoming bitstream [57]. In case data corruption does occur during transfer, the UART protocol includes a parity bit that is used for the receiver to check if the number of ones or zeros is odd or even [57]. All of this is to say that although there is a very basic form of error checking in the UART protocol, it is a far cry from more advanced methods of communication that can not only detect when an even number of errors has occurred in one data frame but can also correct the errors that did happen. Nevertheless, it is still worth mentioning that this protocol can maintain much better data transmission integrity if the option to use oversampling was available and turned on [57]. It was highly desirable to use an oversampling rate of at least 8 or 16 in order to safeguard against the data being transmitted incorrectly.

This project did not need to send a large amount of data through the Bluetooth module and these signal broadcasts happened in bursts with long periods between a bottle refill and reevaluation of the water quality. Therefore, it was not a large concern that the bitstream coming from the microcontroller into the BLE module would utilize the full potential of the Bluetooth data transmission rate. The utmost importance of all that has been discussed in this section is to emphasize the need for preserving the integrity of the data being sent to the smartphone app. Even if the microcontroller properly analyzed a water sample and indicates that it is unacceptable via the status LEDs on the unit; if the data were corrupted over the UART transmission even once, this could potentially confuse a user into thinking that the LED was wrong because the readout on their phone tells them that the water is fine. This might seem like a lot of fuss over too simple of a matter, but even the chance of inappropriately telling a user that non-potable water is safe to drink could have resulted in disastrous consequences.

## Microcontroller Software Implementation

This section outlines the various microcontroller/development board features and peripherals that potentially needed to be utilized throughout the project. Prior to prototyping, it was important to first understand how each type of connection/microcontroller peripheral worked. This then served as a helpful reference for team members writing code later on during more advanced software development. Not only this but outlining how to implement various microcontroller features made it easier to identify sub-goals for software development. Also, compartmentalizing the codebase into smaller, more identifiable, and more manageable pieces helped to smooth out embedded software development.

### GPIO Pins

The most common pin configuration is a GPIO pin. The General Purpose Input/Output’s or GPIO’s in the MSP430 are categorized as ports and pins. In PX.Y, ‘X’ represents the port number, whereas ‘Y’ represents the pin number. Here, the term ‘pin’ is not synonymous with the general understanding of a pin number on an integrated circuit. In other words, there would appear to be duplicate pin numbers here, but the ports are what distinguish the pins and make them all unique. Each port has several registers that need to be written to in order to configure any pins. The register is a bit-wise representation of each pin on a selected port. For example, the 8 bits from right to left in port 1 represent pins 0-7 respectively. The most common way to configure the GPIO’s involves utilizing bitwise operation and/or masking to set and clear certain bits in a register. When performing bitwise operations, it is always helpful to utilize the “BITx” symbolic constants in the MSP430 header file. Then, a define statement can be created at the top of a program, and a more appropriate name can be used for the bit mask.

Configuring a GPIO pin simply involves setting or clearing the respective bit in the PxDIR register. Here, the “x” represents a port number on the microcontroller. Setting a bit in this register will direct that pin as an output, whereas clearing a bit makes the pin an input. All pins are configured as high impedance inputs by default. The next register of importance is the PxOUT register. This is the register used to output a logic level high or low on general purpose output pins. And the last register is PxIN, which is used to read a logic level high or low from input pins. Ports 1-4 on the microcontroller, have interrupt capability. This means that they can wake the microcontroller up from low power mode.

### GPIO Interrupt (LLWU) Pins

When configuring a GPIO pin to be able to interrupt the microcontroller, again another register has to be interacted with. First, the pin must be configured as an input by clearing the corresponding bit in the PxDIR register. Then, it is always helpful to utilize the built-in pullup and pulldown resistors inside of the microcontroller. This can be done by setting the corresponding bits in the PxREN (Resistor Enable) register. Alternatively, this register could remain 0 by default, and external pullup and pulldown resistors can be used. The type of resistor is configured in the PxOUT register that corresponds with the port that contains the desired interrupt pin. Setting a bit will make the resistor a pull up. Whereas, clearing a bit will configure the resistor to be a pull down. The next register, PxIES (Interrupt Edge Select), is used to control the type of edge that triggers the interrupt. A one is a falling edge trigger, and a zero is a rising edge trigger. Next, it is always a good idea to clear the flag (that will normally be raised by the interrupt) so that the interrupt event does occur instantly; this is accomplished by clearing the corresponding bit in the PxIFG resistor. It is always important to remember that interrupts (even if individually enabled) will not execute until the Global Interrupt Enable (GIE) bit is set. This can be accomplished with the function “\_enable\_interrupts”

### Interrupt Service Routines (ISRs)

Fortunately, interrupts are kept relatively simple since they are handled by the hardware. After an individual interrupt event has been configured and enabled, as well as the global interrupt bit has been set, the MSP uses a vector table to handle all of its interrupts. Each interrupt service routine has to have its own programmed function that follows the main function in the code. Because of this, every time an interrupt occurs, the hardware knows which operation to perform next. Some interrupt service routines are shared, in this case it is important to always be able to identify which event triggered the interrupt through the use of if-conditional statements. The hardware will automatically clear the flag after an individual interrupt service routine; however, for shared ISRs, it is the programmer’s responsibility to clear the flag (designate that the interrupt event has been handled).

### Non-GPIO Pin Configuration

Other pin configurations require using a selection register or multiple selection registers to configure non-GPIO functionality. Some of these features may include: PWM output, ADC input, UART Tx, and UART Rx. To do so, it is always best to consult the datasheet to determine which masks need to be used. For example, in order to configure P1.6 into a timer output channel for a PWM signal, the datasheet outlines the following conditions: set bit 6 in P1DIR, set bit 6 in P1SEL1, and set bit 6 in P1SEL0. In order to perform the aforementioned operations, the declarations for each register can be found in the MSP header files. Each pin that is going to have non-GPIO functionality must be configured in this way.

### MSP430 Driverlib

This is an API (Application Programming Interface) that is provided by TI. Application Programming Interfaces typically allow programmers to manipulate lower-level software by using higher level logic and a library of pre-configured functions. It simplifies many microcontroller tasks, as well as abstract register operations. Using this API would help minimize bitwise operations and masking. However, this API did have some complicated software implementation. It required an extra week of troubleshooting to create an empty project with Driverlib by installing the appropriate SDK into Code Composer Studio. Even though it was very difficult to this API implemented, it ended being very worthwhile because it minimized the amount of code to write. It made it easier to configure pins, configure interrupts, check interrupts, as well as many other code features.

### Timers

The MSP430FR6989/5989 microcontroller comes with a copious amount of timing modules. Each timer is capable of running at its own frequency with a multitude of different channels. There also different modes to operate the timer in, such as: continuous mode, and up mode. Channel configuration registers must be written to with a number of different masks in order to configure the timers. The first step to using a timer is selecting the clock. The MSP has certain default clocks, but these can be reconfigured in the code. After the clock has been selected, an input divider is then used to set the period. The input divider can divide the clock frequency by a factor of 1,2,4, or 8. This is used to generate more optional frequency references for the timer. From here the timer will either count up to 65,536 cycles in continuous mode, or another number of cycles set by the user in up mode. This number is limited to a resolution of 16-bits. Channel zero of the timer always specifies the number of cycles (zero-referenced) counted up to during up mode. Each timer comes with different capture/compare registers/channels (that also have external microcontroller output pins). These can then perform logical operations in different amounts of cycles/time.

When trying to create a PWM signal, channel zero sets the period of the signal in up mode. And other capture and compare registers, can be used to change the duty cycle and/or shape of the signal. Consult the documentation for more information about all of the different types of PWM signals that the MSP430 microcontroller is capable of generating.

### Low Power Mode

The MSP430 has many built-in functions for entering low power modes. “\_low\_power\_mode\_x” is the best way to easily enter any low power mode. Here “x” represents a number from 0 to 4. This function automatically enables the selected power mode and activates the GIE bit so that the microcontroller can wake up from sleep. Higher value of low power mode results in a lower sleep current. Additionally, various low power modes have different clocks disabled/enabled. For example, low power mode 3 only uses ACLK. Whereas low power mode 0 utilizes the DCO, SMCLK, and ACLK. For more information, view the family user guide to determine which power mode is correct for device modes of operation. Note: Only use low power mode 4 if an external GPIO pin is able to wake the microcontroller up from its sleep.

## Structural Design/Device Housing

It was necessary to create a structure that could house the entirety of the components while conforming to a convenient form factor for users to easily handle. Along with being easy to handle, the housing needed to be structurally rigid enough to withstand the grip of users while they remove and secure it into the bottle multiple times a day. This casing needed to be screwed into the water bottle in order to securely close the cap in place (or be designed in a manner which could feasibly be converted to such), which would not only keep water inside the bottle but also activate the Reed switch to allow operation of the analyzer and sanitizer features. The consensus among group members was that it would be better to adjoin a custom casing of our design onto the bottom part of the metal cap that comes with the bottle- however, this was amended during senior design two due to difficulties presented by this design in comparison to a mono-material 3D printed housing. While the device would benefit from a reflective base, it was not a worthwhile pursuit considering the other topics that needed to be explored.

While we did intend to have the cap screw into place, and the CAD software chosen can be modified to do so, it was not thoroughly investigate for the prototype (although a commercial product would have threading). The biggest challenge, however, would have been actually finding a way to get a custom-made metallic cap based off of a CAD design, as no team members had experience with CNC machines. This thought process led us to conclude that it would be much more practical and beneficial to utilize the 3D printers that are available in the CREOL senior design lab.

### Original Prusa i3 MK3 (3D Printer)

While no one on the team originally had CAD or 3D printing experience, one of the team members researched the process and utilized online video sharing sites along with the free CAD software application FreeCAD to produce 3D models to fit the desired housing specs.

### FreeCAD

The CAD software of choice was FreeCAD, due to a combination of ethical drives (as it is open-source), monetary draws (as it is free), and practicality pulls (as it allows for complex, precise modelling with an engineering focus). FreeCAD proved an effective software when used on Linux and Windows alike without any discrimination in versions between the two operating systems.

### PrusaSlicer

In order to 'slice' the 3D models (which are in .stl format) into a printable list of commands (.gcode) for the Prusa i3 MK3S printer, we used PrusaSlicer from prusa3d.com. This software is compatible with Windows, MacOS, and Linux operating systems, although the Linux version is not updated as regularly as the more common OS versions. Although we had not used this software before, this software was easy to use and did not require significant research. The difference in regular updates was problematic at one point, as after installing the software on a Windows computer midway through senior design two it was discovered that easy-to-use preset settings allowed for higher-quality prints. While this is not an obvious problem at first, the finer quality led to the device housing's screw holes no longer gripping properly as the surface was smoother. This was remedied by reducing the diameter of said screw holes.

### Prusament (Filament)

There were many different choices of filament that could be used with the Prusa 3D printer, and it was important to choose a material that was optimal for housing design. As previously mentioned, this structure would be twisted by users on a regular basis and therefore it was necessary to select a material that had an acceptably high shear strength. This casing also needed to provide support to the electrical and optical components that rested inside it, so rigidity and strength were also factors that needed to be considered. In order to not interfere with the wireless charging system, the chosen material ideally had a low conductivity, so as not to induce large eddy currents in the casing itself. Finally, it was also highly desirable that the bottle cap would not be heavy because this would have increased the chance that a user might accidentally drop it and damage the components within. Since density is typically correlated with a higher strength and hardness, there needed to be some optimal compromise in the material selection process. It was found that the senior design lab provided PLA filament for students to use in their projects, so this option was selected in order to eliminate the cost of filament. The fill density was generally set to 25%, which appears to be a good compromise of weight and strength.

## Application Software

### iOS App Development

Upon investigation, it was found that the common development environment for creating iOS apps is called Xcode and could have been downloaded for free on any machine running macOS 11 or later. The coding language for iOS apps that was developed by and endorsed by Apple is known as Swift, and one can begin developing an app in this language using the SwiftUI interface within Xcode. Unfortunately, there is limited support for macOS 11 on Apple computers released before 2013 [58], and this was a direct limitation on our ability to use Xcode for iOS app development.

### Android App Development

The official Integrated Development Environment for creating Android Apps is known as Android Studio [59]. This IDE provides a lot of flexibility for developers because it is available for Windows, macOS, and Linux. The official programming language for developing Android apps used to be Java, but it was replaced by a newer language known as Kotlin in 2019 [60]. Kotlin is considered to be easier for beginners to code in compared to Java, and this is a promising feature for this project when considering that none of the group members had experience in programming an app for smartphones.

### Object-Oriented Programming Basics

Since most of this group had not coded in Java or Kotlin before, many of the object-oriented functionalities of these languages were learned through the lenses of students who had hitherto written code mostly in C. The more that online guides and references about programming for Android had been explored, it became increasingly evident that there was a need to investigate some of the most basic concepts and terminology that are used in object-oriented programming such as Kotlin.

The titular feature of object-oriented programming is the software object itself. In terms of coding, an object is a “bundle of related state and behavior” [61]. These states of an object are analogous to a value that describes it, such as name or color, while a behavior is an ability that the object can perform [61]. The various states of an object are stored in fields, which are similar to variables, and the behaviors of an object are “exposed” through its methods, which are like functions [61]. Therefore, an object is a piece of software that has fixed methods through which it can change its states and interact with any other code.

Objects in programming languages like Java are created from a prototype called a class [61]. A class in object-oriented programming is similar to a schema in psychological terms in that creating an instance of a particular class is how we consistently attribute certain properties to objects of that class type. Knowing how a class is defined allows us to see an objects functionality that it can perform and the potential states that it can have.

The concept of inheritance leads to the idea of a higher level of classes known as a superclass. Grouping similar classes together underneath a superclass allows each of these subclasses to inherit common features the superclass that they are all categorized in, and this is done in Java by using the keywords “extends” to reference a particular superclass in the new class declaration [61]. This is a coding concept that is very useful for creating new classes that are based off of a pre-existing superclass so that the programmer can focus on the added functionality that they are including to define their new class.

It has already been discussed that an object “exposes” itself to software via methods that are a part of their definition, but these methods can be identified as an interface [61]. Interfacing with an electrical object in the physical world is typically done through some sort of control panel and defining an interface in object-oriented programming is a way of laying out a grouping of methods that can be called upon in a class declaration via the keyword “implements” [61]. This idea of an interface is a way of standardizing and formalizing the ways in which anything can interact with a defined class.

The final concept that is presented in the Java tutorials being referenced as [61] is a “package”. In Java, the term package refers to a grouping of interfaces and classes that are all related, and this is a way of organizing them [61]. The multitudinous number of classes within Java and Kotlin gives rise to the need for further organization of these pieces of code. The term Application Programming Interface (API) is what programmers call the library that contains packages of classes and interfaces that form the building blocks for any code written in that particular programming language. Now that these rudimentary terms of object-oriented programming have been covered, it was much easier to follow the discussions online about how we proceeded in using a given API to attempt to create the Android app for this project.

### Android BLE App Development

In order to try to create an Android app that could communicate with a Bluetooth Low Energy module, it was necessary to explore some basic terminology and programming design considerations. Without getting bogged down with information that was not necessary for the execution of this project, it became very clear that there are certain terms that needed to be discussed in order to make even the humblest beginning in BLE coding. The term that seems to show up most often in BLE discussions is GATT, which stands for Generic Attribute profile, and this describes the formatting of services and characteristics, as well as the processes for interfacing with these attributes [62]. A GATT Service is a grouping of related characteristics that refer to a particular device feature and a GATT Characteristic is basically a data field that can be edited and read from [63].

The Cypress BLE module will be acted as a peripheral or server that stored the values of water quality within a GATT characteristic maintained on the module. After the microcontroller updates the values of the water quality within the BLE module, the module can send an indication to the smartphone (acting as the central or client) that will tell the client it may ask for an updated value of the characteristic (data field). It was previously explored how the Cypress module could update an acknowledged GATT data characteristic (via UART from the microcontroller) that the smartphone has subscribed to. Since we will be using the acknowledged data characteristic, this should push an indication to the client (smartphone) to say that the data is ready to be sent. An indication is similar to a notification, except that it means that the data packets received by the smartphone will need to be acknowledged in order to guarantee their delivery [63]. This is the general process that will be undertaken in order to transmit water quality data to the smartphone app, but there is still a need to discuss how the Cypress module and the Android app are going to connect in the first place.

The Android BLE guide by Chee Yi Ong [63] specifically recommended creating an app that is designed for Android 5.0 (API 21) or newer, due to the better BLE Application Programming Interfaces (APIs) that are available. It is also explicitly stated within Android Studio that Android 5.0 Lollipop was created with support for Bluetooth Low Energy. Android Studio also indicates that using newer APIs will reduce the percentage of phones that can run the app, so it seems that API 21 will be the best option to use for developing our app.

Looking further into the BLE guide by Chee Yi Ong [63], there is a list of classes that are suggested to be very useful in the process of coding in Kotlin for the integration of BLE connectivity. Reading about these various classes and their uses offers insight into how one can actually code an application to interface with a device via Bluetooth as well as what steps are necessary to successfully find and connect to the device in the first place. The “BluetoothAdapter” class is what represents the Bluetooth hardware on the smartphone itself [63]. Interfacing with an instance of this class can provide the programmer with information such as which Bluetooth devices are bonded to the Android phone along with supplying the capability of start a scan for BLE advertisements. The class that can start a BLE scan is called “BluetoothLeScanner”, and this is provided by the “BluetoothAdapter” class [63].

Once a scan for BLE advertisements has been initiated, it is possible to filter the results based on the Universally Unique Identifier (UUID), device name, MAC address, service data, or manufacturer specific data [64]. It will be necessary to query the Cypress module from the microcontroller via command mode in order to obtain any of these identifying pieces of information in order to let the Android app know which device it should be looking for. If this application was going to be used to connect to one of any smart water bottles being mass produced based off of this project, we would probably filter results based off of the manufacturer specific data. However, for the purposes of designing a prototype device for the satisfaction of this project’s requirements, we will most likely use either the UUID or device name to find and connect to the Cypress module.

Once a BLE scan has been narrowed down based on the known device identifier, the device advertisement will be represented by the aptly named “ScanResult” class [63]. Using the “getDevice()” method we will be able to expose the “BluetoothDevice” handle, which is a class that represents that actual Bluetooth device in the Kotlin/Java code, and this will also enable the application being created to actually connect to this device [63]. The “BluetoothGatt” class is what will be used for interfacing with the Cypress module’s GATT profile so that we may access the services and characteristics of the BLE device. As discussed in the EZ-Serial firmware guide, it will be necessary for the application to subscribe to either the acknowledged or unacknowledged data characteristic that will be exposed after connecting with the Cypress module in CYSPP mode [11). It is the intention of this project to prioritize data integrity over application execution speed, so we will be using the “BluetoothGatt” class in order to subscribe to the “BluetoothGattCharacteristic” of the Cypress module, which will most likely be called “Acknowledged Data”. Finally, the interface that will be primarily utilized for getting callbacks about characteristic indications and reads will be “BluetoothGattCallback” [63].

The term “callback” was not made clear within the previous resource that was being used, so it was necessary to discuss what a callback is. Programming languages like Kotlin use callbacks as one way of dealing with asynchronous communications or subroutines [65]. Basically, when a program is waiting for some function or communication to be completed, it is standard practice to avoid “blocking” the program from continuing to do anything else while it is waiting [65]. Using a callback happens when a function is passed to another function as a parameter with the intention of returning upon completion of the desired task and the reception of a callback [65]. The idea seems to be similar to the concept of an interrupt, where we may continue processing other things in the program while we wait for a callback to inform the program that the previously requested task has been completed and so the program may return to processing the result that it had asked for. Since BLE hosts only send information to a client whenever it is ready to send an updated data packet, it makes sense that a Kotlin program for an Android app would need to utilize callbacks as a way to prevent blocking of the program while it waits for any communication from the BLE.

This section will end by discussing the some of the major warnings that the guide by Chee Yi Ong [63] had laid out in preparation for creating a Kotlin program to handle BLE communications. The first suggestion was that operations should be coded to queue up and only be executed one at a time [63]. Each time a request of any type is made by the program, we will want to wait to receive a callback for verification before allowing the next operation to take place. Also, it was suggested that one should keep in mind that any operation being executed has the chance to fail and that a good program should be able to deal with such failures as they occur [63]. Error handling is a process that is essential to creating code with any level of robustness and having the program to methodically queue its operations would ensure that we are properly processing and receiving all of the data as it is transferred to the app from the BLE module.

### CySmart App

Even though a considerable effort was made to try to make an Android app that would have communicated with our device, it ended up being too steep of a learning curve. Thankfully, the Cypress company from which we had bought our BLE modules already made an app that was able to communicate with our module. This app was available for both Android and iOS. It used the exact same process of discovery, bonding, and characteristic subscription in order to establish the data pipe between phone and module as had been previously discussed. Using the CySmart app can be summarized by the following steps:

* Start app and enable Bluetooth connectivity
* Find device named “EZ-Serial” and select it
* Select “GATT DB” from list of profiles
* Press the only service, labeled “unknown service”
* (Optional) Select Indicate only characteristic, subscribe to indications (Rx Flow)
* Subscribe to notify/indicate from either of the two characteristics listed with write capability

## Version Control Systems

Version control systems (VCS) are a functional upgrade from shared network drives, enabling more effective cooperation among teams on large projects. This is primarily due to their user's ability to easily explore more unconventional ideas by using branches while leaving a 'save point' in the form of a previous version. These systems also have near-total immunity to accidental deletions, as any missing files can be reacquired with minimal effort and panic from the cloud- a significant improvement over the days of using volatile memory and an uninterruptible power supply at the office as storage. That is not to say that the team members are reliant on an active cloud server throughout the workday, as each member also has a copy of the repository on their respective computers. There are several popular platforms for version control including GitHub, AWS Code Commit, and Azure DevOps. All three of these use Git to handle repositories.

### GitHub

GitHub is a popular VCS owned by Microsoft and commonly used by university students due to its compatibility with most integrated development environments (IDE). Users of GitHub can upload files directly to the website, use the VCS's app, Git, or use an IDE with Git built in. Once a repository is created it can be 'pushed' to (files from the user's computer are uploaded), 'pulled' from (files from the website are retrieved), and reverted to previous instances, as the website saves enough information to 'undo' changes between pushes. This ability to revert changes is tantamount for programming, as procedural mistakes can be more easily fixed without the need for redundancy on the user's side.

### Azure DevOps

Azure DevOps, also by Microsoft, expands upon the VCS base of GitHub with additional administrative utilities and training geared towards software development. Its popularity resides in the corporate world more than the student realm although both are used by each sector due to the contrasting origins of the two systems. While heralded as the successor to GitHub, uptake is slow even at Microsoft itself, which uses GitHub for most of its open-source projects [66].

### AWS Code Commit

AWS Code Commit is Amazon's take on a VCS, utilizing Git to manage user-side repositories as well. Its primary advantage is reduced cost for private repositories compared to GitHub, which is not applicable to an open-source project.

### Version Control Summary

Of the version controls explored, none have a strong advantage. AWS Code Commit lacks the administrative features found in Microsoft's contributions, however, making it less desirable. Of the three, GitHub is the most attractive due to greater familiarity with it, and the ease with which it can be linked by most IDEs.

## User Operation

While product durability, effectivity, and sustainability are largely determined by the design and manufacturing process, certain operating practices and incidents can affect the device's functionality in detrimental or even hazardous manners. Some such incidents can be mitigated during the design phase, but others require user cooperation and attentiveness.

### Ice Cube Usage

When researching about UVC and its ability to sanitize water, a question about the presence of ice cubes in drinking water was raised. Would the device user be able to put ice in the water? In order for this to be acceptable, the UV radiation would have to be able to sanitize not only the water, but also the ice cubes. A study conducted by Ladanyi, and Morrison concluded that the UV was able to penetrate at least 19 cm of ice while killing the bacteria. However, penetration depended heavily on the optical qualities of ice [67]. Since the consideration of ice would greatly complicate the process of testing and verifying sanitization, the device user should be strongly advised not to include any ice in the water bottle. Therefore, the project scope will focus on sanitizing water without the presence of any ice.

### Protection from Impact

One major concern for use of this product is be the lack of shock resistance of the unit. Water bottles and the caps that seal them shut are items that can normally withstand being dropped on a regular basis- this is especially true for uses that involve more active and vigorous activities; it is a very common occurrence that users will drop a water bottle either intentionally or on accident. However, the components of this design do not allow for strong impacts, and such shock to the unit could very easily destroy the reed switch or dislodge UVC LEDs. While the final implementation of the FLASC is solid-state and less fragile than the original spectrometer design, it is still prone to part breakage under abuse. Despite the limitation of material selection with 3D printers available and the fact that none of the members of this team had experience with CAD software, the final design is relatively shock-resistant, although some internal components are likely to dislodge or the reed switch will break upon severe impacts. If future versions were to be developed after this initial prototype, creating a more impact-resistant design would be one of the first priorities due to the aforementioned danger of such shock.

### Water Temperature Range

The bottle content's temperature will, of course, fluctuate- it is unreasonable to expect users to repeatedly fill their bottle with water that is a specific temperature, let alone a unique temperature set by the project development team. The temperature of the bottle's contents directly impacts all system functions to a certain degree, although it affects the optical components and battery most of all. The battery, as with all electrical storage components, will not tolerate boiling temperatures.

To that end, hot water (that is to say, water above body temperature) should not be used in the device. Hot water would also negatively impact the efficiency and durability of the optical components, causing the laser diode (which uses the cap's metallic structure as a heat sink) to have greatly reduced power, potentially causing a failure of the analyzer. With the design of the analyzer in the final state, this would result in false positives rather than the significantly more dangerous false negatives (falsely indicating a clean sample).

Cold water is not a significant issue for the battery as industry sources claim that lithium-ion batteries experience a ~10% performance reduction at water's atmospheric freezing point [68]. The laser diode will experience a decrease in characteristic wavelength which may have proved problematic in the spectrometer implementation, but which should not be an issue in the water quality analyzer (although excessively cold water may cause a false negative). An additional concern lies in the laser diode's increased quantum efficiency at lower temperatures, which can potentially cause damage to the device by sustaining greater optical output than the semiconductor can withstand. This increase in power was handled by simply reducing the operating intensity at room temperature.

# Project Management

The project management section covers the project's cooperative management strategy for designing and programming the device.

## Software Development

Software development in a team environment necessitates administrative control- specifically a shared codebase with backups, task management with issue reporting, and proper programming practices.

### Shared Codebase

As discussed in section 9.8.5, a version control system is an effective way to share project files across multiple devices and users. The shared code base utilized GitHub to sync its various contributor's source files, using Git to establish copies of the project repository on each member's computer. Git allows new files to be pushed to the server-hosted repository where they are available for use by other users. Prior to programming sessions, contributors should pull the server-hosted repository (which updates their repository with any changes).

### Issue Reporting/Assignments

Part of the administrative process is reporting difficulties and issues encountered during programming, and division of labor. While GitHub has an issue reporting system, the team did not utilize the feature as the number of users modifying any given code (generally no more than 2) meant that direct communication over SMS was simpler. That said, larger teams may find this a useful addition to their toolkit. The error reporting on GitHub can be accomplished using the repository's "issues" tab, which is akin to submitting a support ticket. Contributors can leave notes and status updates as issues to keep software-related issues in proximity with the repository. This system also avoids scheduling and issues and miscommunications arising from using other forms of reporting.

### Proper Programming Practices

While the project's source code was intended to strive towards inheritance over composition and to break large tasks into 'helper' functions which perform a single task, a disconnect in object-oriented programming experience made this an unfeasible goal.

## Virtual Breadboard Prototype

Embedded team members all had the same TI development kit and created their own individual breadboard design. The codebase accounted for all device features. However, some features were implemented for each team member’s breadboard prototype, while other more limited/expensive development features were only on one team member’s breadboard. An example of this, was the Bluetooth module development kit. Dean purchase this and implemented it into his breadboard prototype. So, even though Ryan and Matthew did have it, the same codebase was used to account for all device peripherals. A micro-pin table was also utilized so that every prototype was virtually identical and therefore implementable with the same firmware.

# Power Distribution Design

The elements to charge the battery were bought as premade modular components that had their own small PCBs to be connected in stages. This began with the Universal Qi Wireless Receiver Module by Adafruit, which wirelessly magnetically couples with a store-bought Qi compliant power transmitter. The wireless power receiver was connected to an Adafruit Micro-Lipo Charger for LiPoly Batteries, which helps to manage the proper charging of the battery. This charger was then connected to the battery 2,500 mAh Lithium-Ion Polymer Battery, which was also purchased from Adafruit. Once the battery has been charged, the power that it provides needs to be regulated so that the components are receiving the proper voltage and current. The targeted voltages and maximum current demands of each major component have been summarized in Table 28. Current limiting diodes are used for the laser diode due to its steep I-V relation, which makes small variations in voltage result in large changes in output power and input current. The diode's weakness, namely their sharp drop-off in time-space, is mitigated by pulsing the laser on only when needed.

Power Distribution Diagram

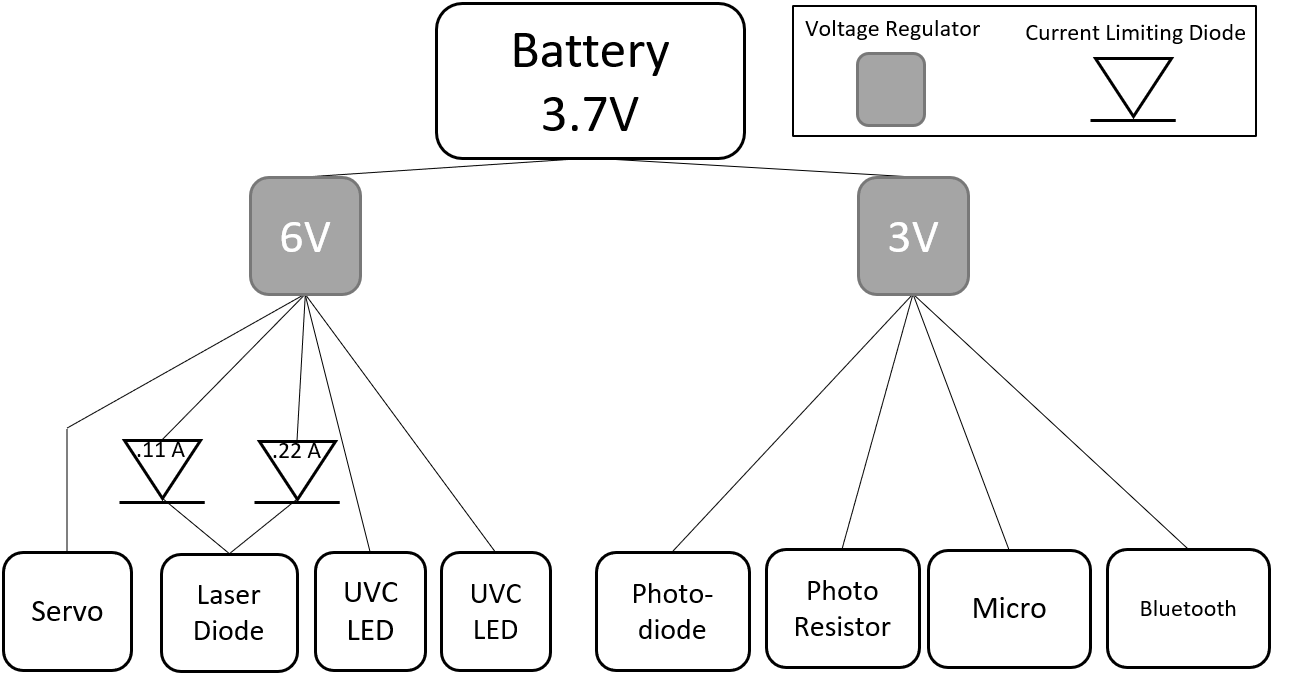


Figure 20: Power Distribution Diagram

Summary of Voltage and Current Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Component (Part Number) | Desired Voltage | Maximum Voltage Ripple | Max Current |
| Microcontroller (MSP430FR5989) | 3.0 V | As small as possible for ADC operation. Include noise bypass capacitors for VREF (10 µF and 470 nF in parallel) | 2.7 mA (at FREQUENCY: fMCLK = fSMCLK = 16MHz) |
| BLE Module (CYBLE-013025-00) | 3.0 V | 100 mV | 28 mA (during BLE transmission) |
| Servo Motor (MS18) | 6.0 V | Not Specified | 90 mA |
| UVC LED 1 () | 6.0 V | Not Specified | 100 mA |
| UVC LED 2 () | 6.0 V | Not Specified | 100 mA |
| Green Laser () | 3.7 V | Not Specified | 250 mA |
| Phototransistor () | <30 V | Not Specified | Not Specified |

Table 28: Summary of Voltage and Current Requirements

We then used the data from Table 28 to generate energy usage requirements for a single cycle of both sanitation, analysis, and BLE data transfer of the FLASC device. The power conversion factor that was used in order to obtain current draw estimates from the battery were calculated by using a simplified power equation of Pout = Pin \* η. The efficiency values were obtained from the data sheets of the regulators; the 6V regulator data sheet was produced by TI WEBENCH. It can be seen below in Table 29 that the total power consumption estimate from the battery at nominal voltage is two orders of magnitude less than the capacity of the battery that we chose. This provided us with a wide berth for our power availability throughout the project.

FLASC Component Energy Usage Estimate

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Component (Part Number)** | **Output Voltage** | **Power conversion factor (step up/down and efficiency)** | **Max Current** | **Time Run for One Cycle** | **mAh consumed from 3.7V Battery** |
| **Microcontroller (MSP430FR5989)** | 3.0 V | (3V / 3.7V) /(.73 efficient) = 1.1 | 2.7 mA (at FREQUENCY: 16MHz) | 3.25 min = 0.054 hours | 0.16 mAh |
| **BLE Module (CYBLE-013025-00)** | 3.0 V | (3V / 3.7V) /(.73 efficient) = 1.1 | 28 mA (during BLE transmission) | 1 sec = 0.00028 hours | 0.0086 mAh |
| **UVC LED 1** | 6.0 V | (6V /3.7V) /(.85 efficient) = 1.91 | 100 mA | 3 min = 0.05 hours | 9.55 mAh |
| **UVC LED 2** | 6.0 V | (6V /3.7V) /(.85 efficient) = 1.91 | 100 mA | 3 min = 0.05 hours | 9.55 mAh |
| **Spectrometer Green Laser** | 6.0 V | (6V /3.7V) /(.85 efficient) = 1.91 | 250 mA | 2 sec = 0.00056 hours | 0.27 mAh |
|  |  |  |  | **Total** | **19.54 mAh** |

Table 29: FLASC Component Energy Usage Estimate

## Voltage Regulators

Between all of the optical components, the servo motor, the microcontroller, and the Bluetooth module, there were a multitude of different voltage and current requirements for this project. It was decided that including more than two voltage regulators in the project design would not only be wasteful in terms of project spending, but it also would require an inappropriately large amount of space on the custom PCB which was less than three inches in diameter. The voltage of the Li-Po battery is 4.2V at full charge, nominally 3.7V, and stops discharge at 3.0V when it is depleted (12). Therefore, these were the voltage input specifications used for the TI WEBENCH Power Designer online.

The first regulator that was created was the 3.0V DC-DC converter that powers the microcontroller, the BLE module, and provides biasing for the spectrometer photodiode. Inputting the voltage range of our battery selection with the desired output voltage of 3.0V, there were many balanced designs presented at first. Choosing an output current for this regulator was difficult, because the current demands for a regulator that is already stepping down the voltage are expected to be very small. It might seem like one would simply make the output current to be about 40 mA, since the components being connected to it were not expected to draw any more than this during peak operational periods. However, it was found that regardless of reducing the scale of expected output current, the designs presented by WEBENCH still showed the same efficiency drop-off curve as the current draw approaches zero. Since there was no efficiency improvement to be had by designing for a lower amperage, it seemed sensible to design the regulator with components rated for current comparable to the output of the other regulator; the calculation of which is discussed later in the respective section. When initially narrowing down the various design options, the main consideration for this regulator was the output voltage ripple. The datasheet for the MSP4305989 clearly recommends keeping the voltage source noise to a minimum in order to achieve the most accuracy in the measurements made by the Analog to Digital Converter (ADC) (6).

Investigation into multiple different components throughout this project made it clear that sometimes obtaining specific parts for a design can lead to complications caused by the supply chain. It also was desirable to keep the custom PCB as simple as possible for multiple reasons, but in particular we wished to minimize the amount of space that was required to implement the power conversion design. A considerable amount of space in the cap casing was already to be taken up by the battery and the hardware that was purchased for properly charging it, so in order to be mindful of the size of the cap casing that we designed, it was important to ensure that we also kept the BOM area as small as possible. However, in comparison of the two most balanced designs that are shown below in Table 30 listed by the switching IC used for each respective design, it was decided that reducing the number of components to be included while maximizing the efficiency would make the most sense for our project design. Although there was not nearly as much power consumed by the digital devices and one photodiode when compared to the components that were connected to the 6.0V regulator, it was still better for heat and battery life considerations to prioritize the efficiency of the converters.

Comparison of 3.0V Buck-Boost Converter Designs

|  |  |  |
| --- | --- | --- |
|  | TPS63024 | TPS63020 |
| BOM Area | 102 mm2 | 62 mm2 |
| BOM Cost | $1.42 | $1.43 |
| BOM Count | 7 | 10 |
| Efficiency | 96.1% | 93% |
| Frequency | 2.5 MHz | 2.4 MHz |

Table 30: Comparison of 3.0V Buck-Boost Converter Designs

Next, we needed to select a design for the 6.0V voltage regulator for the rest of the components. Again, many options were provided by the TI WEBENCH Power Designer, but the two designs that were most balanced in the desired qualities are listed below in Table 31. Selecting an appropriate output current for this design definitely had more concrete constraints that could be calculated. First of all, it was the intention of this project to code the microcontroller in such a way that would allow us to operate each of the major current consumers one at a time. This allowed us to minimize the peak current draws from the battery which will not only preserve its state of health, but also reduce the heat generated per second (assumed to be proportional to the square of the current due to I2R losses). Thus, it seemed that we would be drawing the max output current from this regulator while operating the green laser for the spectrometer. However, there was one more design factor that needed to be considered when choosing an output current and that was the recommended maximum output current of the battery. The Adafruit website for purchasing the selected battery suggests that it is best if the current draw is limited to 1A from the battery. It was therefore time to make a simple conservation of power equation that could estimate the current output of a boost regulator when the input is 1 Amp. The intention was to select a design with an efficiency well over 90%, yet these listed efficiencies are the maximum possible efficiency that can be obtained as the current draw approaches the designed upper limit. Thus, it seemed reasonable to calculate with an estimated efficiency of 85%, since the lower limit listed on many of the efficiency curves shown by WEBENCH were around 86-87%. Taking into consideration the lower limit of the battery voltage as it approaches the end of its charge, which is 3.0V, we were left with a power equation as shown below:

Pin x η = Pout

Vin x Iin x η = Vout x Iout

Iout = (Vin x Iin x η)/Vout

Iout = (3.0V \* 1A \* .85)/(6.0V)

Iout = 0.425 A

With this output current calculation finished, the value was used for the TI WEBENCH power designer to generate the initial set of design options to choose from. Many balanced designed were initially sorted out of consideration due to the emphasis on selecting a regulator with an efficiency as high as possible. Some of the highest efficiency designs had at least 20 different components just for the single voltage regulator and these were not ideal selections in the context of this particular project.

Comparison of 6.0V Boost Converter Designs

|  |  |  |
| --- | --- | --- |
|  | TPS61372 | TPS61089 |
| BOM Area | 112 mm2 | 91 mm2 |
| BOM Cost | $2.09 | $2.37 |
| BOM Count | 11 | 14 |
| Efficiency | 91.4% | 95.4% |
| Frequency | 1.49 MHz | 493.58 kHz |

Table 31: Comparison of 6.0V Boost Converter Designs

## LDO Linear Regulator Addendum

After much deliberation on the effects of signal noise from the switching regulator, it was finally decided that an LDO linear voltage regulator would be more appropriate for supplying the 3.0V rail with power. Even though a large portion of time and effort went into the pursuit of minimizing the noise from the 3V switching regulator, the final report created by the TI WEBENCH online tool showed that the output voltage of the device would still have had a peak-to-peak voltage ripple of 30 mV when the output current dips below 170 mA. Considering that we did not intend to ever draw this much current from the 3V rail and how sharply the curve of the Vp-p increases as the amperage drawn decreases, this means that we simply could not have expected to achieve a stable reference voltage for any of the analog components in the process of converting optical intensity into a stable input voltage for the microcontroller to read.

Therefore, it was decided that we would rely on a Low Dropout (LDO) linear voltage regulator. Looking online for in-stock components that are RoHS compliant and sorting by increasing price, there was one model of regulator that quickly stood out as meeting all of the needs for this regulator design. The MIC5365 LDO voltage regulator by Microchip Technology / Micrel is rated to have an output voltage of 3.0V, a dropout voltage of 155-310 mV at max rated current that is 150 mA, and it accepts an input voltage between 2.5-5.5V (17). The datasheet also recommends using a 1 µF capacitor at the input and output pins, respectively, that would help stabilize the voltage supplied the 3.0V rail. The maximum line regulation of this device is 0.3% and has a typical regulation of 0.02%, which means that we can expect for our 3.0V rail to be held within 0.6 mV of whichever nominal value (±2% from 3.0V) our regulator is suppling to the digital and analog loads. When compared to the ripple that was inherent in the design of the switching regulator, this device would hold our 3V rail 50 times more stable with the line regulation. This provided the highest quality results of the ADC when taking readings of the water quality, which was extremely important considering the vital nature of determining whether or not water is potable.

# Embedded Hardware Design

This section will outline all of the processes, considerations, and iterations that went into designing the schematic and printed circuit board. Once technical investigation and a general project overview were completed, it was time to start organizing and interfacing all of the various electrical components. This process began with studying data sheets, designing various circuits, selecting pin configurations for the microcontroller, creating breadboard protypes, and utilizing EAGLE’s schematic software. From here, the board layout was created based on the schematic in EAGLE. Then the board was ordered by a contract manufacturer and assembled overseas. The entirety of this design process required careful balance and consideration between software and hardware capability. When designing the printed circuit board, the software was created alongside the EAGLE schematic. This allowed for quick and easy adjustments between the software and the hardware/pin configurations during the board design process. The hardware design will be broken up into several subsections. This helped to better organize and plan out the hardware design process.

## Utilizing EAGLE Software

Other PCB software could have been utilized, but team members for this project had the most experience with EAGLE. This preliminary experience cane from the pre-requisite Junior Design course as well as a Co-op/internship. This software allowed for quick and easy schematic creation, as well as board layout creation. In addition to this, EAGLE keeps track of all components/footprints and generates a bill of materials that can then be given to the board manufacturers. EAGLE also generates the GERBER files that board manufacturers used to create the printed circuit board. Thus, the software was an invaluable tool that allowed the team to efficiently create a printed circuit board for the project.

EAGLE was first utilized for fast and streamlined schematic creation. During this time, it was imperative to select appropriate symbols, footprints, and devices. The symbol is the schematic representation, the footprint is the board layout representation, and the device merges the two together in EAGLE. Fortunately, a lot of integrated circuits and microcontrollers already have user-created EAGLE libraries that can be found online. The device simply has to be inspected to make sure dimensions and layer rules match, and then imported into a project library. Some device modifications were performed in EAGLE. For example, every single components’ silkscreen was modified to create a universal outline (in mil) for all PCB components, as well as the pin names in the symbol. The schematic is the best referential point for how the board functions. Later on in the testing and coding phase, it proved to be a very useful and valuable tool. Therefore, the schematic needed to be easy to follow, and have logical organization and compartmentalization of all of the various device sub-systems. One way to improve organization was to take advantage of busses in EAGLE. So not every connection had to be made, and yet the busses made it clear which microcontroller pins connect to each sub-system. Likewise, the more resemblance there is between the schematic and the board layout, the easier it is to analyze and test the product.

The next part of EAGLE design involved using the previously created schematic to place components and route traces between components on the circuit board. When designing the board, there were several important factors to keep in mind. Some components had to be placed in certain locations. An example of this would be the buttons and indicator LEDs that had to be visible to the user from the exterior of the bottle cap. It also was a wise idea to make sure that the linear voltage regulators were placed on top of a heat sink plane (bottom and top of board). As this protected it from excess voltage that could possibly damage it. And, that the switching regulator plane was designed in accordance to manufacturer recommendations for a highly efficient design. It desirable to create an even ground pour between the top and bottom of the board to minimize cross talk between PCB layers. In general, the board layout needed to be easy to follow, logical, and closely match the schematic. It was also a good idea to create sub-sections throughout the board that compartmentalize device sub-systems. This made it easier to identify device systems later on. Although it wasn’t feasible by the end, it was preferrable to fit all components on the front of the board. This could have eliminated the need for a back-side PCB stencil and decrease overall costs. However, some optical components required being attached from the back side of the board via external 2-pin connectors, based on the final housing and physical layout of FLASC.

### Microcontroller

Since this component was the central unit that interacted with just about every part of the circuit board. It appeared to be a logical starting point for the hardware design process. First the pinout/configurations of the microcontroller needed to be heavily analyzed and scrutinized. Less common pin functionalities were assigned to device subsystems, then followed by more common typical GPIO functions.

### Device Sub-system Connections

In order to design the pin configuration for the MSP430FR5989 microcontroller it was first important to consider the different sub-systems and their input/output to and from the microcontroller. This both provided better organization throughout the design process and ensured that all components had been considered and nothing was overlooked. It would have been very troublesome to start assigning pins for the microcontroller and overlook components that had to be forcibly implemented into the design later. This section outlines these numerous systems and also shows which type of connections they needed to have with the microcontroller. Latter parts of the design process (such as microcontroller pin configuration) could not be performed until every feature had been analyzed and broken down into specific types of connections for interfacing with the microcontroller. This section is similar to outlining the ideal microcontroller which was performed under technical investigation. However, it now has more specificity since the microcontroller has been selected and more detailed information was determined about the microcontroller’s features and pinout. For the optical components, Table 32 covers the sensor connections required for the spectrometer/Water Clarity system and Table 33 establishes the outputs and inputs needed for the Sanitizer. The electrical systems and processes (Bluetooth module, microcontroller flashing/Battery voltage, and indicators) are summarized by Table 34, Table 35, and Table 36 respectively.

Water Clarity and Obsolete Water Quality

|  |  |
| --- | --- |
| Servomotor Or Stepper motor (both obsolete) | PWM timer output signal. 3 GPIOs for direction, sleep, and 1 extra. |
| Detector (phototransistor) | 1 Analog I/P to the Microcontroller |
| Green Laser | PWM Signal to pulse laser enable signal. MOSFET to switch on circuit. 2 Different power settings. |
| Water Quality Button | GPIO LLWU |

Table 32: Water Quality Sensor Connections

Sanitization

|  |  |
| --- | --- |
| UVC LEDs (Quantity of 2) | GPIO enable pin and MOSFET to switch on circuit. |
| Reed Switch | LLWU/Interrupt Pin on microcontroller (GPIO P1-4) |
| Sanitization Button | GPIO LLWU |
| UVC LED Feedback--Photoresistor | Analog Input |

Table 33: Sanitization Connections

Bluetooth Module

|  |  |
| --- | --- |
| Pairing Mode Button | Hold down Analyze button for 2 seconds. |
| UART | Tx and Rx |
| Flow Control | CTS and RTS (2 GPIOs in total) |
| Mode Selection | CSYPP (GPIO) |
| Connection | GPIO |
| Data Ready | GPIO |
| Low Power Mode | LP\_MODE, GPIO |
| Low Power Mode Status | GPIO |

Table 34: Bluetooth Connections

Microcontroller Flashing and Battery Voltage

|  |  |
| --- | --- |
| Programming Header | For microcontroller flashing |
| Battery Read Voltage | Analog I/P to microcontroller |
| Battery Read Enable | GPIO |

Table 35: Microcontroller Flashing Connections

LED Indicators

|  |  |
| --- | --- |
| R/G/Y Combined LED | 3 GPIOs |
| Blue Sanitizing LED | GPIO |

Table 36:LED Indicators Connections

### Pin Configurations

Pin configuration is crucial when designing an embedded product. Microcontrollers today have a number of different ways that they can be configured based on different applications. It is important that pins with multiple features are designated one specific job. This then requires implementation from a software perspective to set the job/task for each pin. The MSP430FR5989 has a lot of pins, and therefore it was relatively quick and easy to create a pin configuration for this project due to an excess number of pins with various capabilities. It is always more important to map scarcer pin functions first, and then more common ones later.

#### Pin Configuration 1

This pin configuration focused on creating a fully functioning prototype (while utilizing the Launchpad’s features), and later applying the same configuration to the final 5989 microcontroller. In order to create a breadboard prototype for the project, it was first necessary to analyze the MSP430FR6989 (100-pin model) pinout. This was because it was the microcontroller that came on the MSP-EXP430FR6989 Launchpad--which was being utilized for the prototype breadboard design. At the time, it seemed very likely that the microcontroller on the final printed circuit board would follow the same layout. The main reason for this was that the code from the breadboard prototype would be easily transferable to the final product. Although the pin numbers are different on the 5989, the pins and ports still have the same functions, and the same header file can be used, so it is pretty easy to convert between the two microcontroller pin configurations.

First, the UART Tx and Rx pins were assigned. Next the analog inputs and PWM signal pins were assigned. The last assignment involved GPIOs (both with and without the ability to interrupt the microcontroller). It was important to ensure that GPIOs were assigned to pins on the launchpad that have external jumpers to plug into the breadboard for prototyping. Pin assignment was performed in this order so that scarcer project functions could be configured first, then followed by more prevalent/common project functions (such as GPIOs). Also, other microcontroller connections were made on the launchpad; however, only pin configurations related to I/O project functions were considered in this section. Table 37 below outlines the MSP-EXP430FR6989 Launchpad pin configuration for the breadboard protype project:

Development Board Micro-Pin Configuration



Table 37: Pin Configuration 1

This configuration would have certainly made the most sense if the final product was on the Launchpad. However, the schematic for the MSP430FR5989 microcontroller would have become much more convoluted. Traces would not have been routed to the microcontroller in logical busses, and lots of vias would been used to interconnect every component correctly. It was also noticed later on that this configuration utilizes pins and ports that don’t even appear on the final microcontroller. So, this would also have led to more software overhead since the pins would have needed to be reconfigured. Although this was a viable option, it still seemed wise to create a pin configuration more focused on the final product.

#### Pin Configuration 2

Another option for pin configuration involved first inspecting the MSP430FR5989 pin-out and then basing the configuration off of that. This method inevitably resulted in a more complex breadboard prototype due to connections no longer being made based on the launchpad itself. However, this also resulted in a cleaner board routing for the final project, so it was worth it. The configuration still needed to work with the development board, so there were some important considerations that needed to take place. The first major consideration that needed to take place was the fact that all connections (with exception of the launchpad buttons) needed to have an external header on the MSP-EXP430FR6989 Launchpad. So, first the BoosterPack header on the launchpad was inspected and it was determined which pins/ports had an external header. The main reason for this, was that the development board needed to be able to connect to the prototype breadboard. Table 38 below outlines the header connections available on the development board:

Development Board BoosterPack External Headers

|  |  |
| --- | --- |
| Port 1 | Pins 3,4,5,6,7 |
| Port 2 | Pins 0,1,2,3,4,5,6,7 |
| Port 3 | Pins 0,1,2,3,6,7 |
| Port 4 | Pins 0,1,2,3 |
| Port 5 | None |
| Port 6 | None |
| Port 7 | None |
| Port 8 | None |
| Port 9 | Pins 0,1,2,3,4,5,6 |

Table 38: Development Board External Headers

After figuring out this information, it was important to only select pins that have an established BoosterPack header/jumper connection. From here, the pin configuration process began to take place. The very first pins configured were the Launchpad buttons. Since these buttons have a fixed port/pin number on the development board, this configuration had to match on the printed circuit board. Then the UART Rx and Tx pins were selected. From here the PWM signals were selected. They both had drastically different timing/duty cycle requirements, so they were configured to be on separate timers. They also needed to be on timing channels greater than zero since this was the only way to trigger a PWM event. Channel 0 on the timing modules was used for setting the period in up-mode. Next, the analog inputs were selected. They were kept relatively close to each other during the pinout; however, the launchpad buttons got in the way from routing all of them in one bus of traces to the microcontroller. Then, all of the GPIOs with interrupts were chosen—that meant these had to be on Ports 1-4. And lastly, the remaining GPIOs were configured in a logical fashion. E.g., all of the Bluetooth GPIO connections were kept very close to each other in the pinout. Table 39 below outlines Pin Configuration 2 which was obtained by the end of Senior Design 1:

Microcontroller Pin Configuration 2



Table 39: Pin Configuration 2

### Final Pin Configuration

This section will highlight the final pin configuration that was obtained by the end of Senior Design. This final version shows the inclusion of the stepper motor, as well as unused pins configured as pulled high inputs to decrease device power consumption. Every row highlighted in light blue was implemented in the final design, whereas every row highlighted in green was unused and configured to decrease power consumption. Shown below in Table 40 is the final pin configuration that was used for this project.

Final Microcontroller Pin Configuration



Table 40: Final Microcontroller Pin Configuration

### Spy-Bi-Wire Programming

Programming the MSP430FR5989 on the custom PCB required some header pins to allow for easy access to the microcontroller. The TI Launchpad that contains the 6989 chip was able to be used for programming an external target by utilizing the eZ-FET portion of the board. The jumpers that connect the eZ-FET isolation block to the rest of the Launchpad were removed in order to expose the header pins that allowed for the 5989 to communicate with a PC for programming. This programming was done with a simplified form of JTAG communication configuration that TI calls Spy-Bi-Wire (6). In this interface the header pins for Vcc, Vss, SBWTCK (clock signal), and SBWTDIO (data in/out) of both the Launchpad (eZ-FET side) and the 5989 target (custom PCB) were connected temporarily via jumper wires which allowed for a program to be delivered from Code Composer Studio onto the microcontroller used for this project. There still existed a discrepancy between the 3.3V Vcc pin that resided on the Launchpad and the 3.0V DVCC1 pin that was supplied by the battery and linear regulator locally. Therefore, we did not connect these power supply pins together but it still was our intention to add header pins on the custom PCB for DVSS1, DVCC1, SBWTCK, and SBWTDIO to ensure that we were able to access all of the pins when it came time to program the device. An alternative involved only connecting the grounds of the devices and supplying power from the battery. If the DVCC1 and DVSS1 pins were not needed for programming because the device was being powered locally, then this at least gave us easy access to measure the supply voltage that was being supplied to the 5989. This measurement was an important part of testing to ensure that the voltage supply for the 3.0V rail was being regulated within the limits that were put forth by the regulator datasheet.

## Preliminary Schematic

Once the pin configuration was complete, it was time to work on the schematic in EAGLE. The schematic design process involved dividing the project into smaller more manageable sections and working on one section at a time. During this time, component electrical characteristics (such as resistances, voltages, and currents) were selected, and devices were imported into EAGLE.

All resistances needed to follow EIA-96 guidelines. The design had to have enough flexibility so that another board revision was not necessary; so, any future implementation already needed to be accounted for in this revision. For example, if a bypass capacitor or external pull-up/pull-down resistor may have been necessary, then this part was included on the board (but left un-populated). For the project, external pull resistors were not necessary. This was because the microcontroller ensured that all port pins were high impedance with Schmitt triggers prior to pin configuration (and before enabling I/O functionality). So, it seemed that internal pull-up and pull-down resistors would work for the entirety of the project--without causing any false interrupt events. It was still a good idea to always clear the flag before enabling an interrupt in the code to further prevent unintended interrupt events from taking place. This design also required further investigation into how external components would interface with the board. As a placeholder, temporary jumpers were used to represent these external connections. Later in the schematic design, more reliable connectors were selected for interfacing with each external component.

It was also important to update the packages of each device and make sure that they were correct (according to technical documentation) before creating the printed circuit board. For routing nets on the schematic, busses were utilized so that the schematic connections could be simplified and easier to follow. This section outlines the full Rev. 1 schematic, as well as explain the design process that went into creating each device sub-system.

### Voltage Regulators

This section outlines the schematics for the 3V and 6V switching regulators, as well as the final +3V LDO regulator. For more information about the design of these regulators, see Section 11. In Figure 21 and Figure 22, the 3V switching regulator and the 3V LDO regulator are shown, respectively. Whereas, in Figure 23 the +6V boost regulator is shown.

#### +3V Regulators

The initial switching regulator was previously thought to be a suitable +3V supply component for the design. So, it was initially included in EAGLE, and is therefore now provided in this document as a reference. However (as stated previously), it was decided later on that a LDO would be a better choice due to analog voltage reading requirements for the final product.

+3V Switching Regulator Schematic (Obsolete)

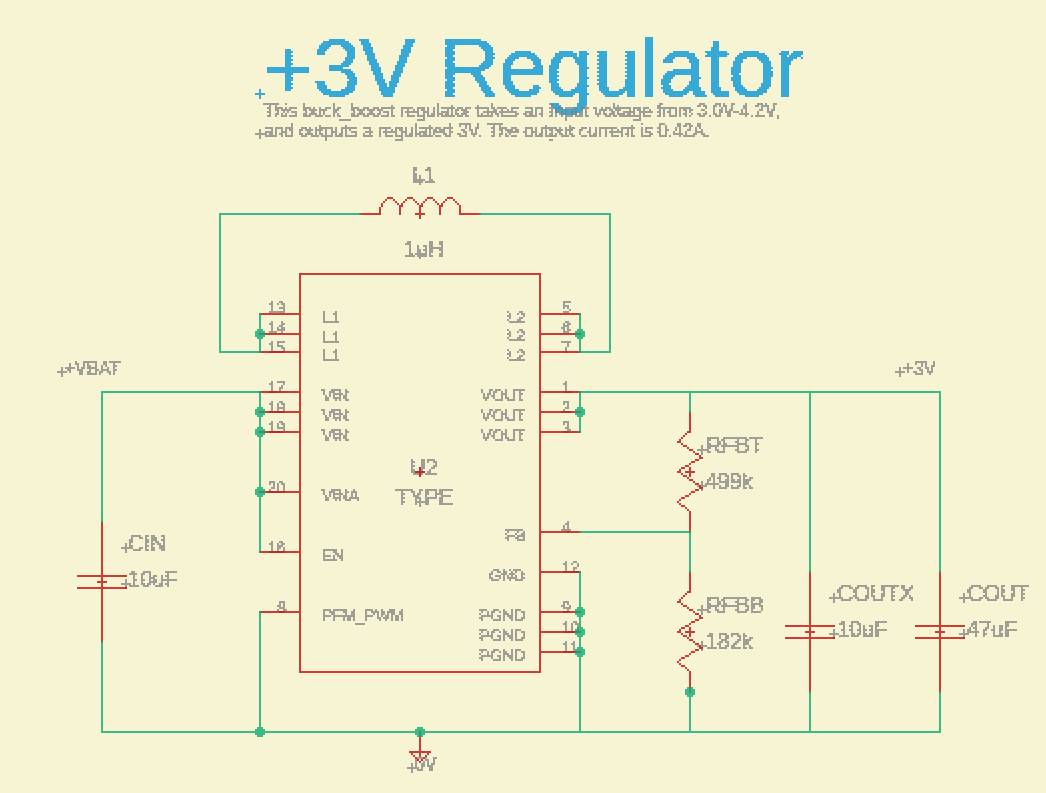


Figure 21: Preliminary 3V Switching Regulator Circuit

The original +3V LDO schematic utilized a voltage regulator that was unfortunately unavailable in the marketplace. Due to this reason, a different LDO regulator was selected for the final Rev. 1 design.

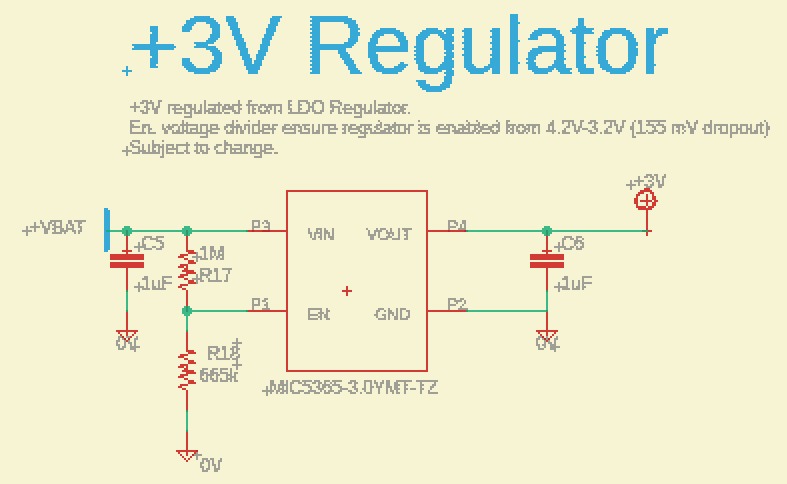
Original +3V LDO Regulator Schematic (Obsolete)

Figure 22: Preliminary +3V LDO Regulator Schematic

#### +6V Switching Regulator

The initial +6V switching regulator circuit again utilized an integrated circuit that became out of stock around the approximate time that boards were to be ordered. For this reason, the +6V regulator was also redesigned using WEBENCH for the final Rev. 1 design.

+6V Switching Regulator Schematic (Obsolete)

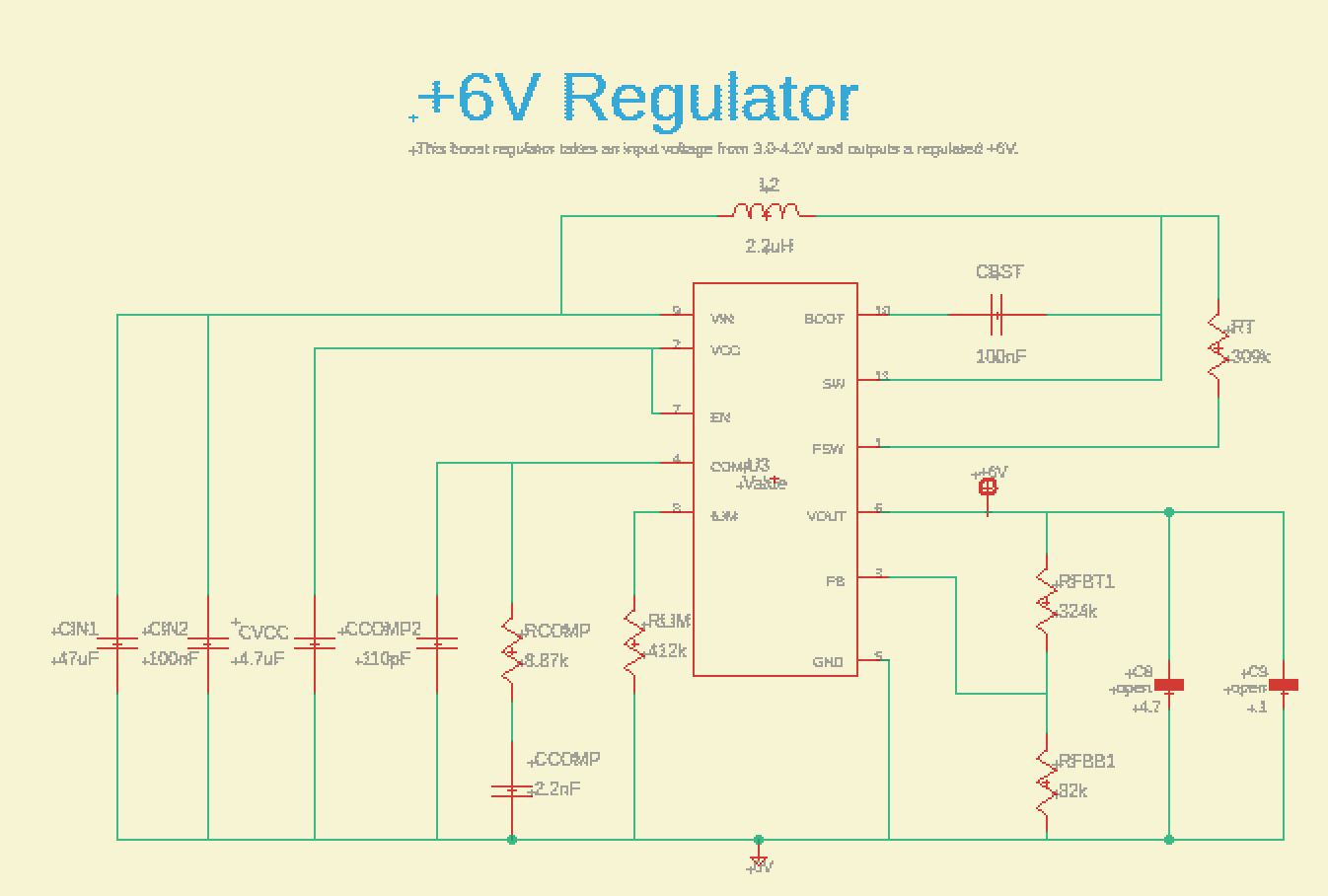


Figure 23: Preliminary +6V Switching Regulator Schematic

### Microcontroller

The microcontroller schematic follows the pin configuration provided in the previous design section. All necessary GPIO connections were routed to a bus which could then connect to any other section in the EAGLE schematic. This made the schematic a lot easier to both route and follow. The microcontroller symbol did organize the pins by their port/pin designation according to TI. So, this would have made all individually routed micrcontroller connections on the schematic much more complicated to look at it. However, it was previously stated that the microcontroller needed to be configured so that connections made logical sense based on their designated pin number. Shown below in Figure 24 is the microcontroller schematic.

Microcontroller Schematic

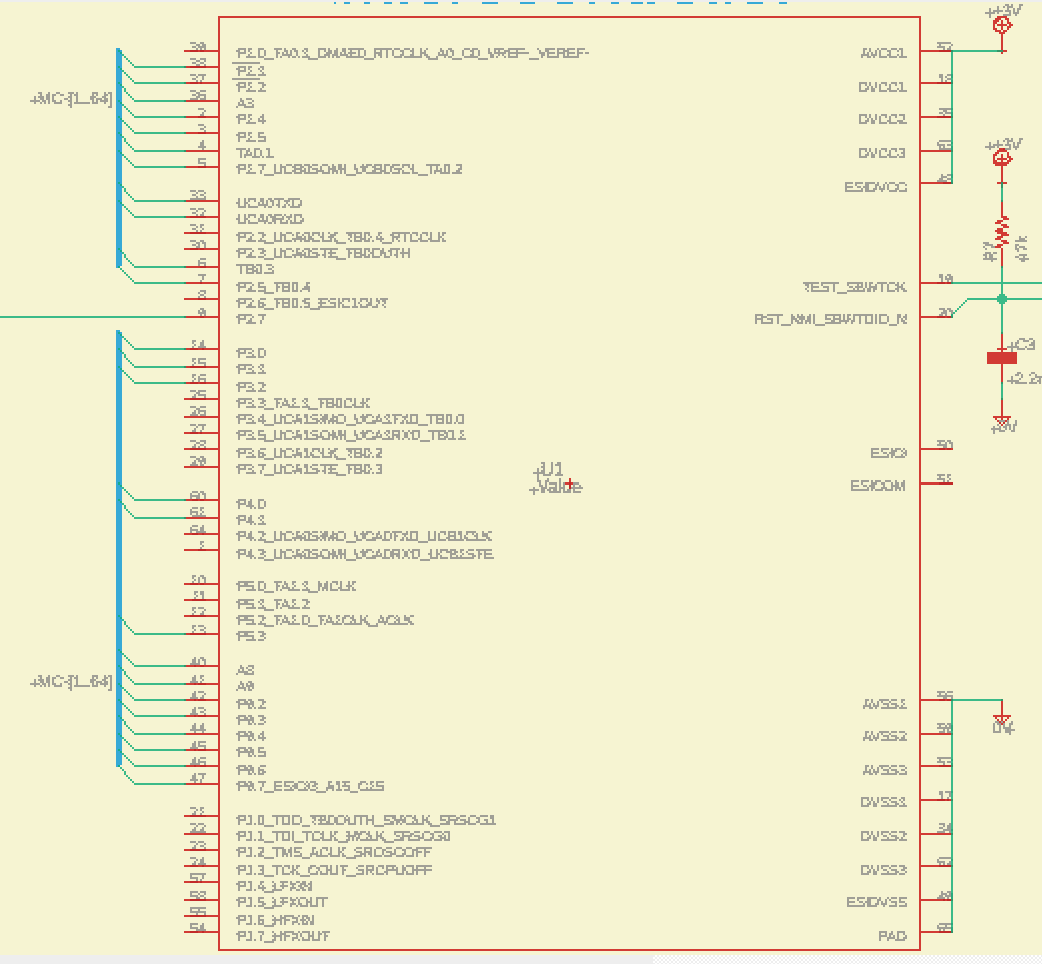


Figure 24: Preliminary Microcontroller Schematic

It was also recommended by TI to include power system decoupling capacitors (1uF and 100 nF in parallel) for each VCC/VSS pair. These connections were made in the final board layout. But they were loosely represented in the figure shown above. Originally more noise/bypass capacitors were considered to be added to the microcontroller schematic (VREF, VeREF). After further consultation, it was concluded that those additional capacitors were not necessary for the design. In Figure 25 below, the power decoupling schematic is shown.

Power Decoupling Schematic

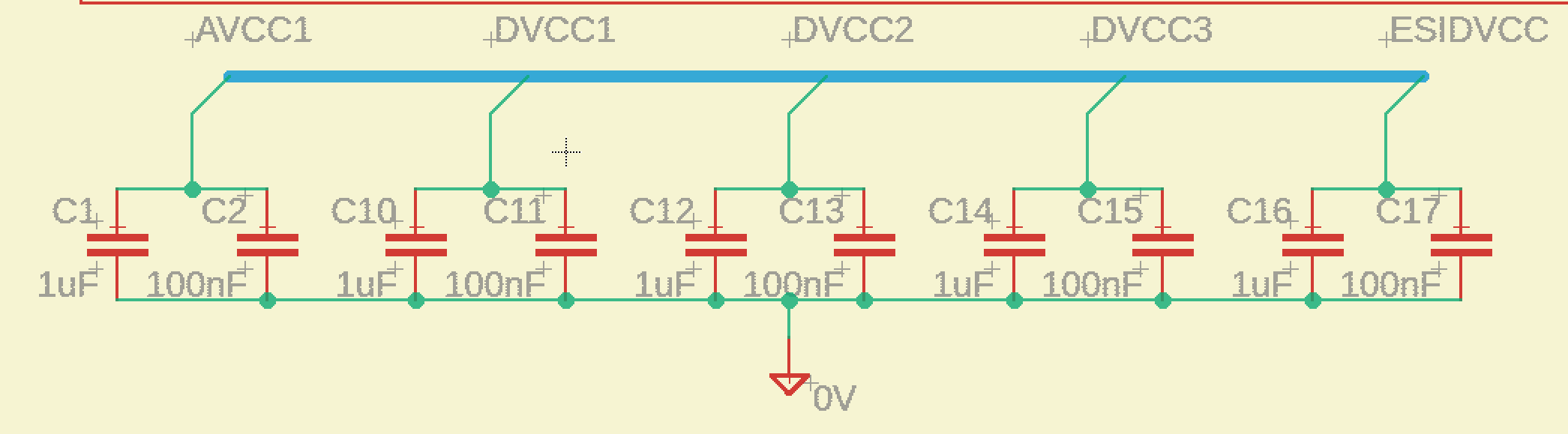


Figure 25: Preliminary Power Coupling Schematic

### Battery Circuit

The battery charging circuit and battery are connected externally via Adafruit sub-boards. However, a known JST PH 2-pin connector was selected to plug the battery into the board. From here, it was desired to have a power switch for the user to toggle device power on/off. Initially this was done directly through a switch; however, it was difficult to find a switch that could support enough current. So, the switch was then attached indirectly using a P-Channel MOSFET to turn the device on/off.

From here, it was also desired to have a method to read the battery analog voltage so that crude battery life estimations could be reported to the user. This was done by enabling a N-Channel MOSFET which then asserts a voltage drop across R5 so that the P-Channel MOSFET will pass the battery voltage to a voltage divider. This voltage divider then divides down the battery voltage so that it is never larger than VCC for the microcontroller. This is the only way to read this voltage, and it required later multiplying the result by the inverse scale in the software to obtain an estimated battery voltage reading. Alternatively, the direct ADC value itself could have also been used. Shown below in Figure 26 is the battery circuit schematic.

Battery Circuit Schematic

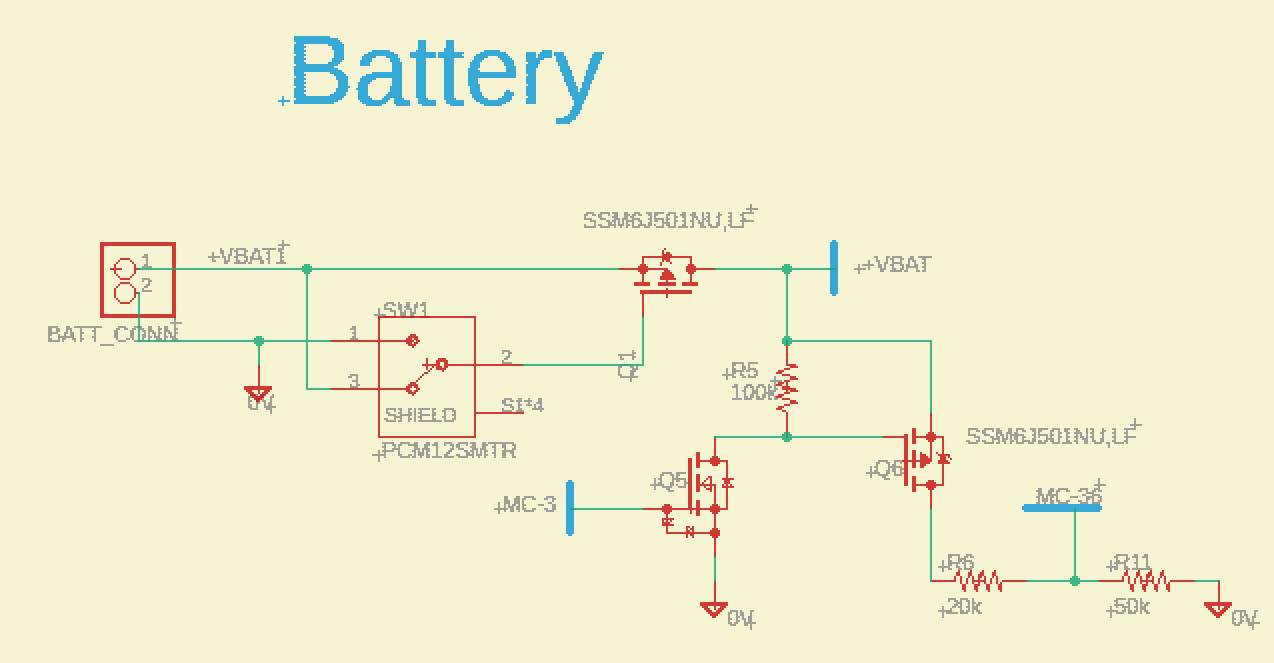


Figure 26: Preliminary Battery Circuit Schematic

### Indicator LEDs

The indicator LEDs circuit mainly involved LEDs connected in an active low configuration to the microcontroller. The red, green, and yellow LEDs were all a part of one device. Whereas the blue LED was a separate component. All anodes of this circuit were connected to the +3V rail. The cathodes were all connected to their own microcontroller pin with a current limiting resistor. In Table 41 below, the calculations for LED operation and voltage/current approximation are shown. In Figure 27 below, the schematic for the indicator LED circuit is shown. Note: Forward voltage and currents are approximations based on component datasheets that were subject to change depending on LED brightness in the final printed circuit board.

Indicator LED Electrical Characteristics



Table 41: Indicator LED Electrical Characteristics

Indicator LEDs Schematic

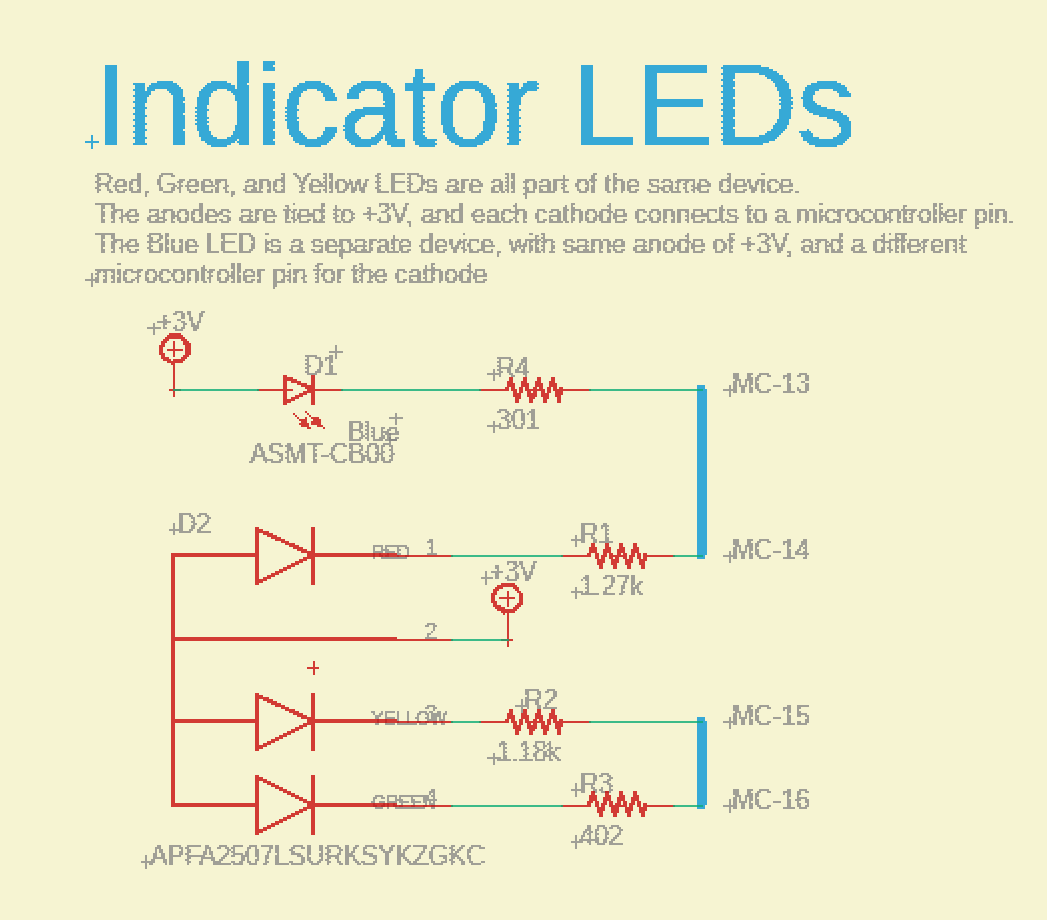


Figure 27: Preliminary Indicator LEDs Schematic

### Push Buttons

The push buttons were used for activating various device modes such as: Water Quality, Sanitization, and Bluetooth pairing. A button press triggers each button’s individual event; however, the analyze button can be held down to activate Bluetooth export mode. Each button connects to 0V, so it is an active low configuration. Internal pull-ups were utilized for each button signal. Each button also required its own shield to be connected to 0V. In Figure 28, the schematic for the push buttons is shown.

Push Buttons Schematic

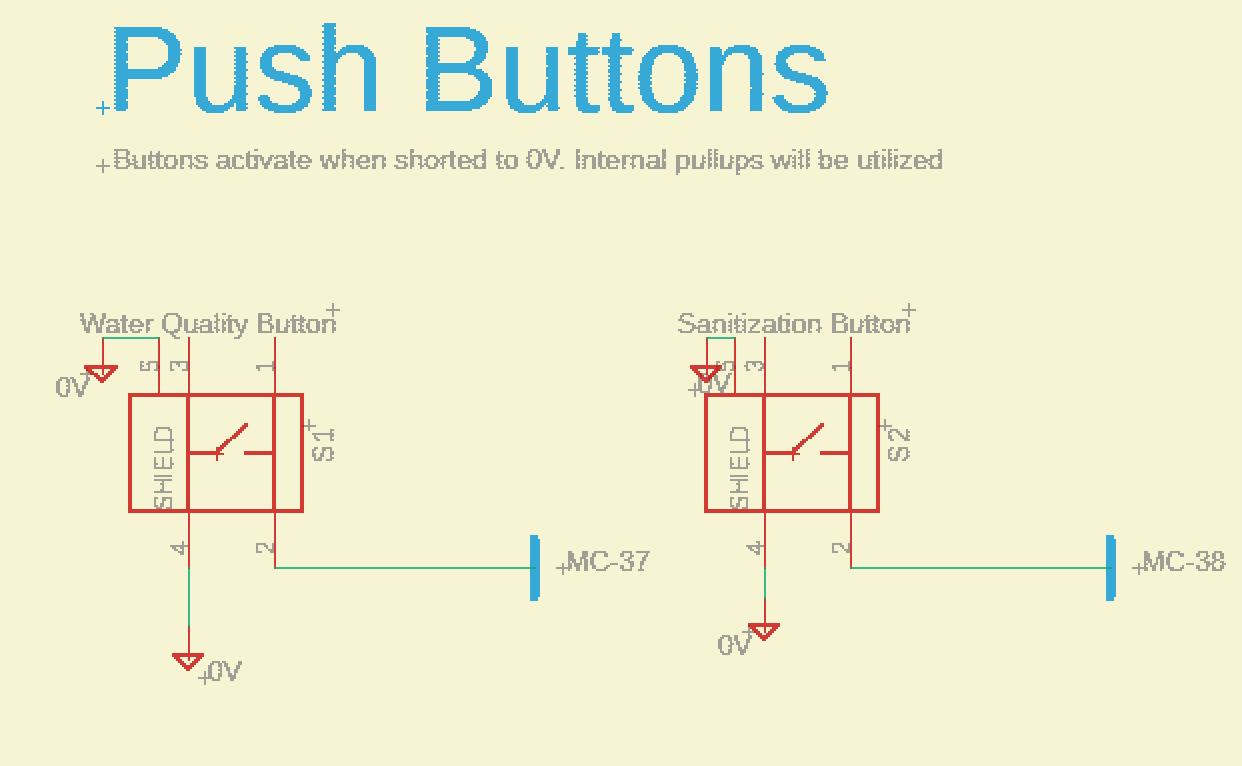


Figure 28: Preliminary Push Buttons Schematic

### Laser Diode and Analog I/Ps

The first iteration of the laser diode circuit attempted to utilize current limiting diodes for current regulation. The laser diode was connected externally and required two different modes of operation. In low power mode, it drew 110 mA through the two current limiting diodes seen on the left. The 1k resistor in parallel with these diodes was going to be used to adjust the current of each current limiting diode. When high power mode needed to be enabled, Q2 needed to be turned on so that an additional 100 mA was drawn to operate the laser diode at a total current of 210 mA. Q3 was used to provide the PWM Enable signal that pulsed the laser diode on/off. Lastly, 0 ohm resistors were included at the gate of each MOSFET in case the response time of the MOSFET switching needed to be modified at a later time. Note: Another version of this circuit was considered to be implemented that used a bipolar transistor to increase the regulated current of one current limiting diode. This would have eliminated two CLDs in the final design.

Each optical analog input had its own necessary biasing voltage and microcontroller pin for reading said voltage. The photodiode had a negligible current draw; however, the photoresistor was much higher therefore it was provided an enable signal to the microcontroller. The connection for each was initially thought to be a through-hole connection to the PCB. But this did eventually change to an external connector. Shown below in Figure 29 is this section’s schematic.

Original Laser Diode and Analog I/Ps Schematic (Obsolete)

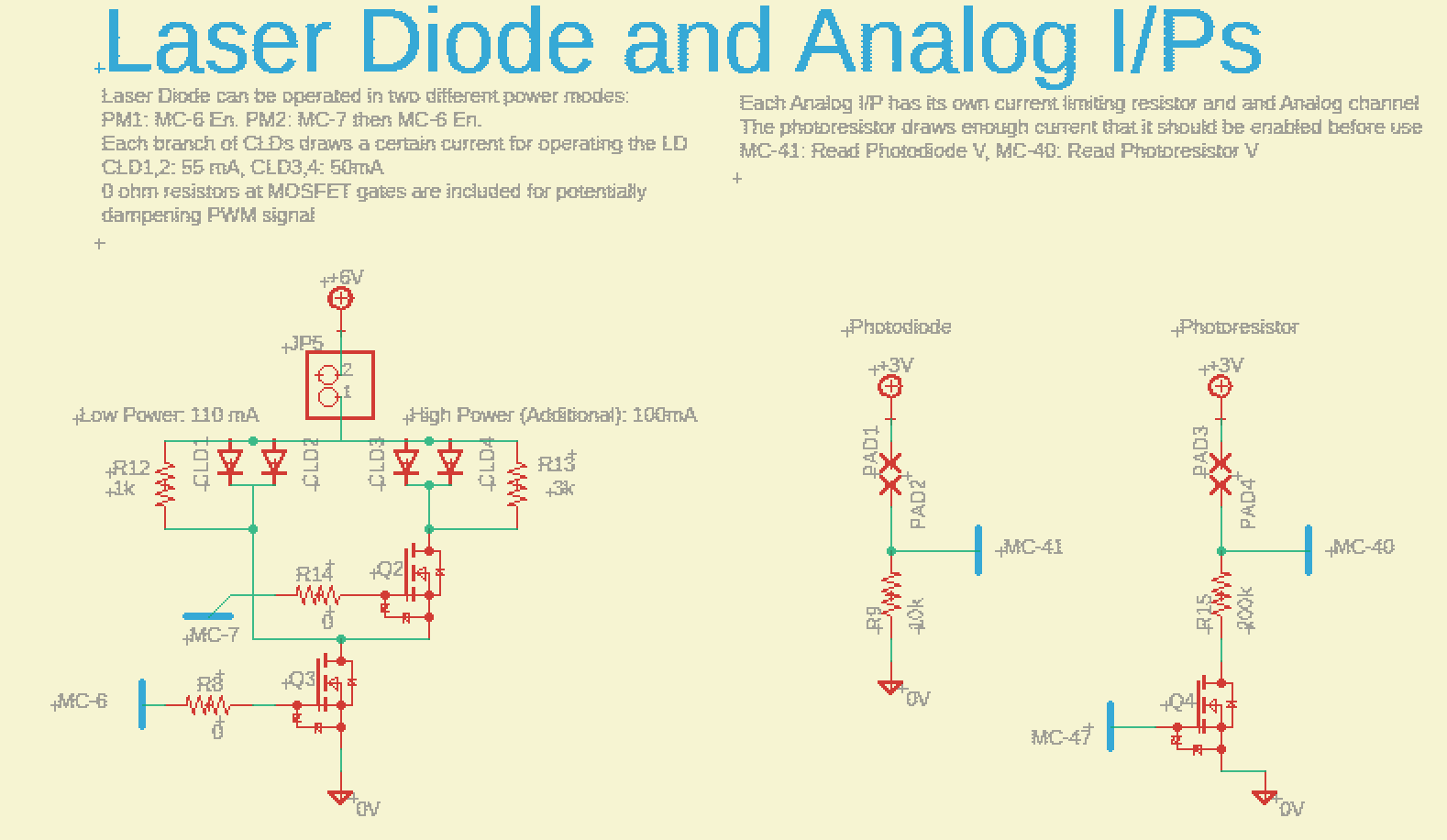


Figure 29: Preliminary Laser Diode and Analog I/Ps Schematic

### UVCs and Servomotor (Obsolete)

The UVCs are connected in parallel externally, and they have their own enable signal that connects to the microcontroller. The servomotor was also external and required connections to 0V, 6V, and a PWM signal. Since the motor was an inductive load, an open resistor across the servomotor was included in case the instantaneous current from turning off the motor required a discharge path. The servomotor also had its own enable pin on the microcontroller. Shown below in Figure 30 is the UVCs and Servomotor schematic.

UVCs and Servomotor Schematic (Obsolete)

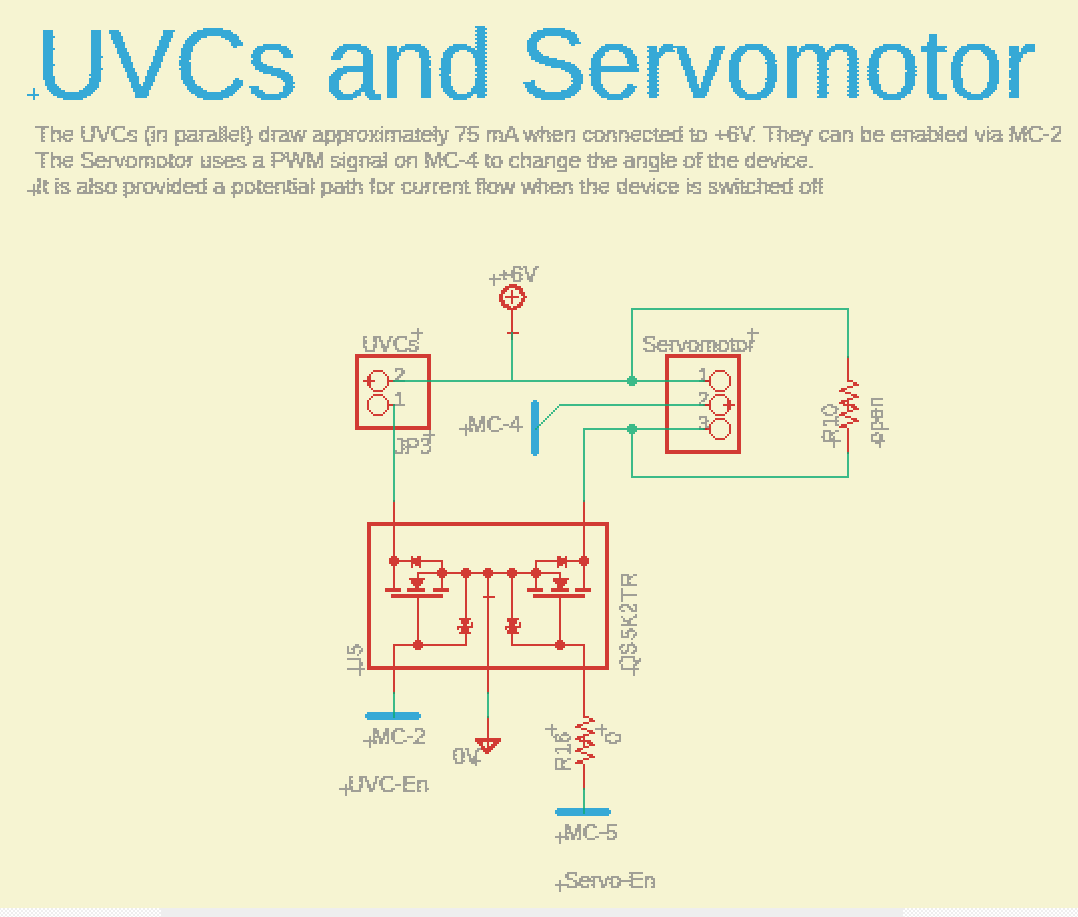


Figure 30: Preliminary UVCs and Servomotor Schematic

### Bluetooth Module

The Bluetooth module requires power connections, and GPIO connections defined in the pin configuration table, Table 8-1 of the Bluetooth Evaluation Module guide, and Section 15.2 of this document. There also is an XRES pin which is recommended by Cypress to be attached to a 0.47uF capacitor to hold the pin low for approximately 50 ms during device power-up. Shown below in Figure 31 is the Bluetooth Module Schematic.

Bluetooth Module Schematic

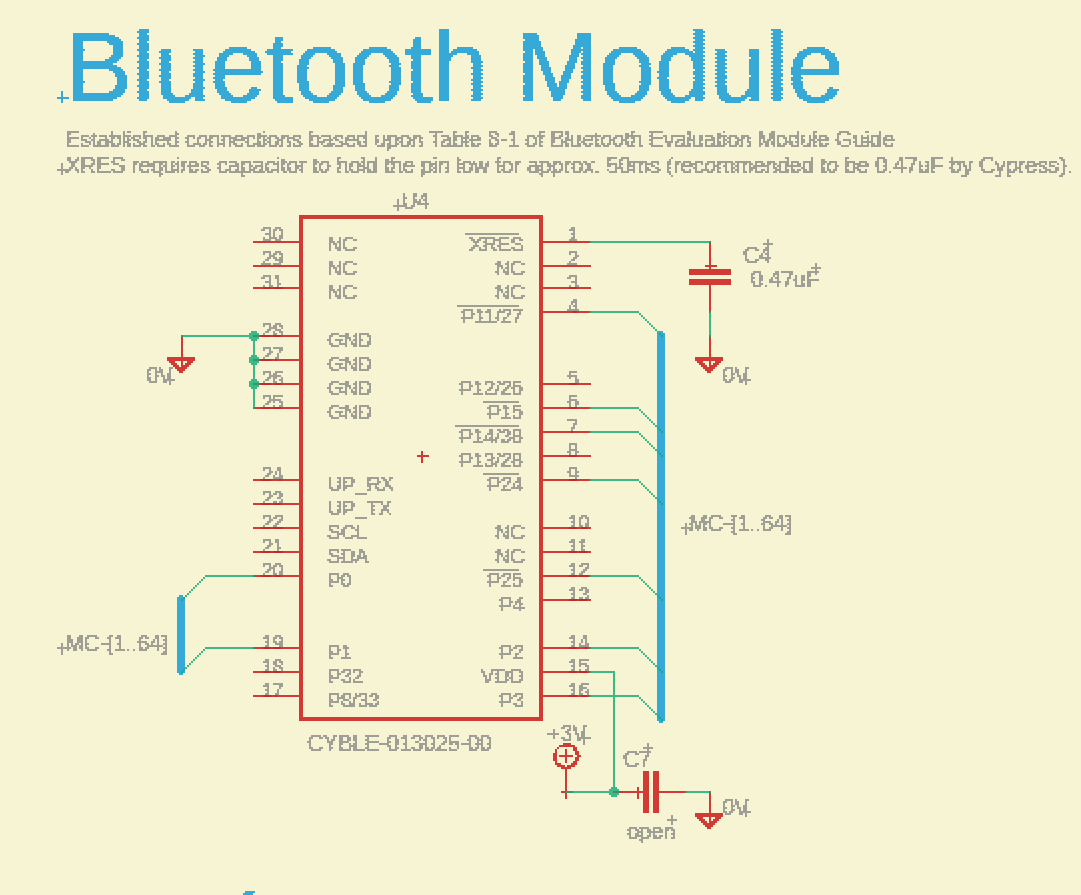


Figure 31: Preliminary Bluetooth Module Schematic

### Reed Switch

The reed switch was connected externally and is normally open. It only triggers an event when shorted to 0V (active low). This meant that a normally closed reed switch needed to be purchased for project implementation. This pin was pulled high internally by the microcontroller. Seen in Figure 32 below is the reed switch schematic.

Reed Switch Schematic

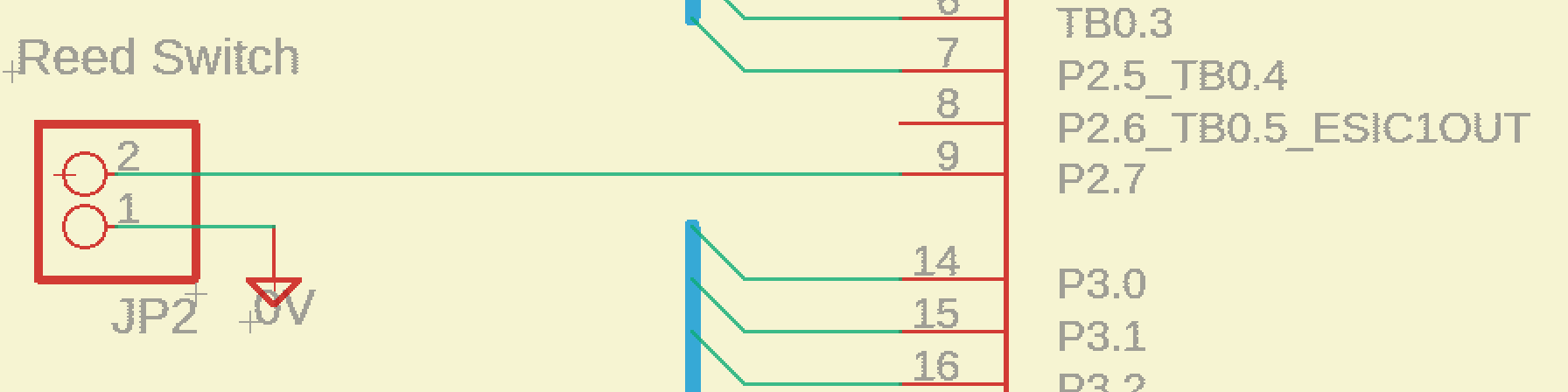


Figure 32: Preliminary Reed Switch Schematic

### Microcontroller Flashing/Programming Header

At the time, it seemed that a 5-pin programming header would suffice. More connections were eventually desired at a later time. This served as a placeholder for how to flash the microcontroller. This schematic is shown in Figure 33 below.

Programming Header Schematic (Obsolete)

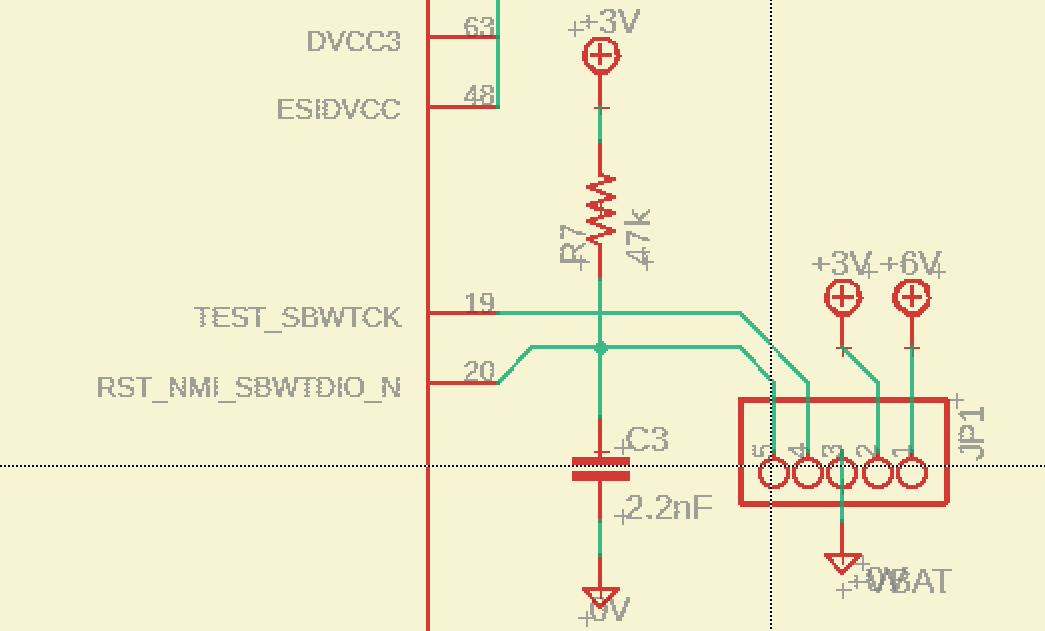


Figure 33: Preliminary Programming Header Schematic

## Final Rev. 1 Schematics

This section will cover the final Rev. 1 schematic that was ordered from PCBWay. If a particular subsection of the design is not explained in great detail throughout this section, that means that the implementation has been carried over from the preliminary schematic. (See Section 12.2 above) Even unchanged schematics will be presented in this section in an effort to make the document easier to follow. The final Rev. 1 schematic in its entirety can be found at the end of this section.

### Voltage Regulators

#### +3V LDO Regulator

This circuit was redesigned and simplified by using a general layout and a MCP1702-SOT23. The chip easily met input/output voltage and current requirements for the project. A constant current draw of 50mA would yield a dropout voltage of 150 mV. This is both in-line with the previous design, and manageable for the project. Code could have also been implemented using the battery read circuit so that the device shut off before the regulator reached the dropout threshold. The design was also reduced down to three components which meant it would take up less space on the PCB. Its simplified design uses a 1 µF capacitor on the input and output rails for noise filtering. Shown below in Figure 34 is the final schematic for the +3V LDO regulator.

+3V LDO Regulator

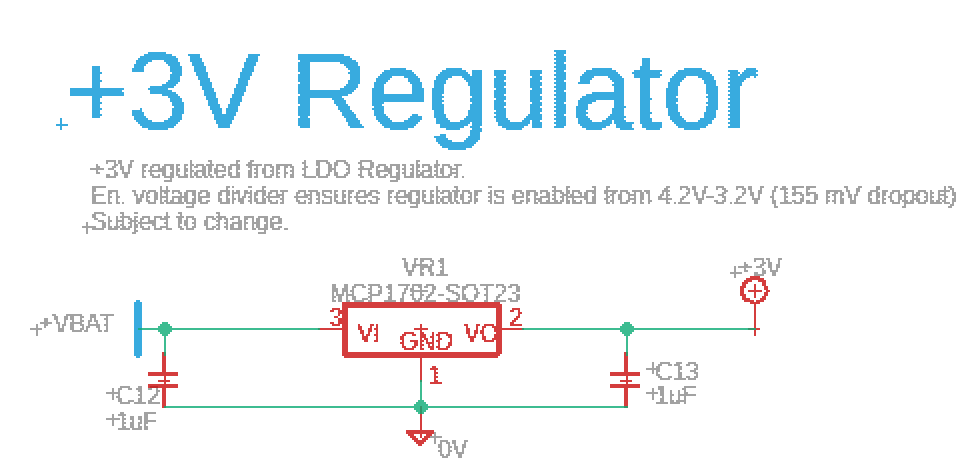


Figure 34: +3V LDO Regulator Schematic

#### +6V Switching Regulator

The final Rev. 1 +6V switching regulator schematic was created using an in-stock +6V regulator IC and WEBENCH. Shown below in Figure 35 is the schematic for the final +6V switching regulator circuit. Since the switching IC that was needed for the initial design of this regulator was unable to be obtained, it was necessary to go back into WEBENCH and choose a design that uses an IC that is in stock. This design ended up being fairly similar in terms of board space, number of circuit components, and the cost of components. However, the main drawback from this redesign was the drop in peak efficiency from 95.4% down to 85.6%. However, It was not expected that this drop in efficiency would have prevented the battery from powering the components for a sufficient number of cycles. Instead, just that it might have led to increased heat waste within the cap casing during operation.

+6V Switching Regulator

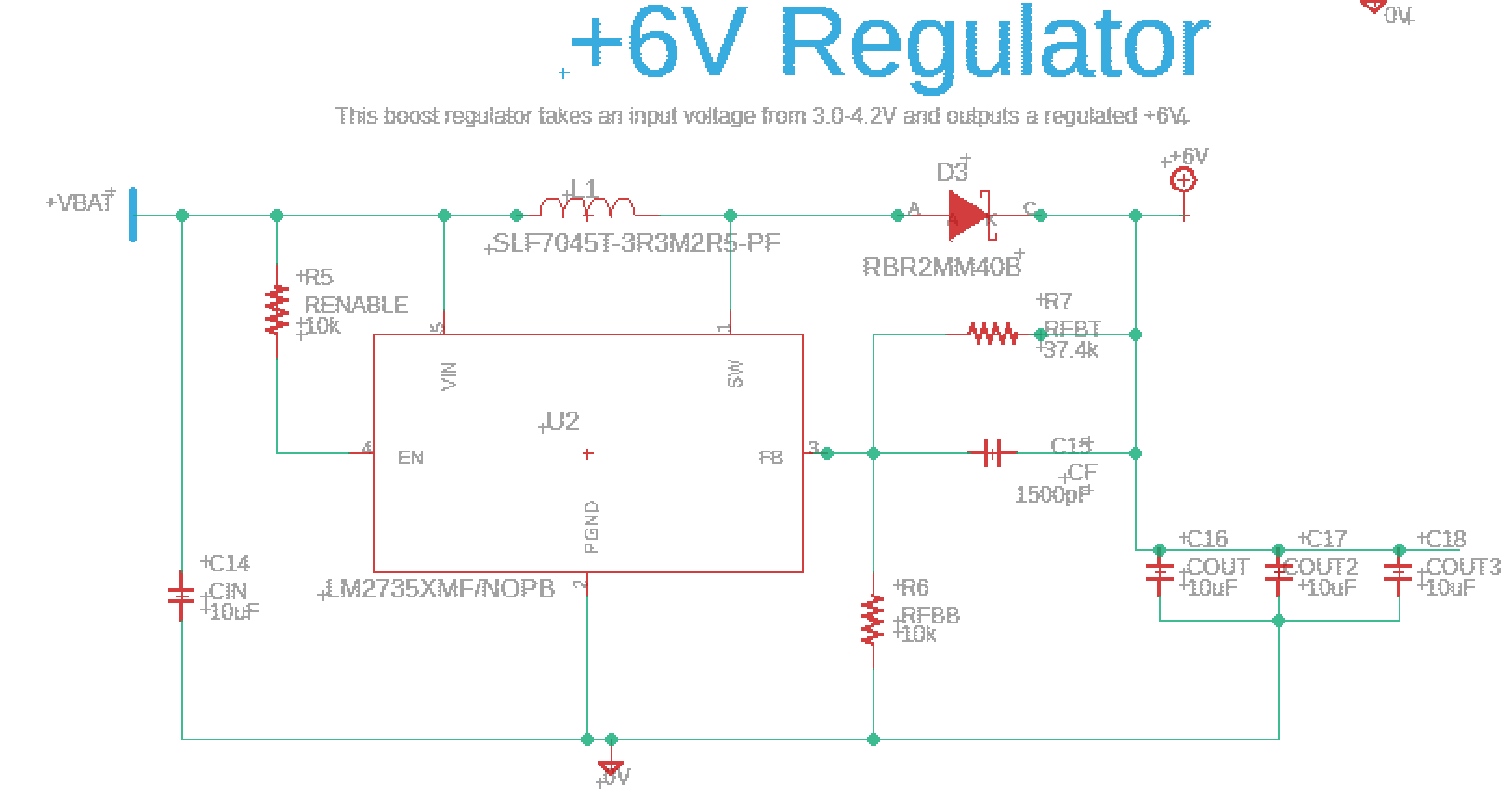


Figure 35: +6V Switching Regulator Schematic

### Microcontroller

Most of the microcontroller’s GPIO connections remained the same, aside from a few changes and additions for new components/device features. For noise coupling, it was ensured that every analog and digital VCC pin had two capacitors in close proximity that connected to ground. Shown in Figure 36 below is the final Rev. 1 connections for the microcontroller.

Microcontroller Schematic

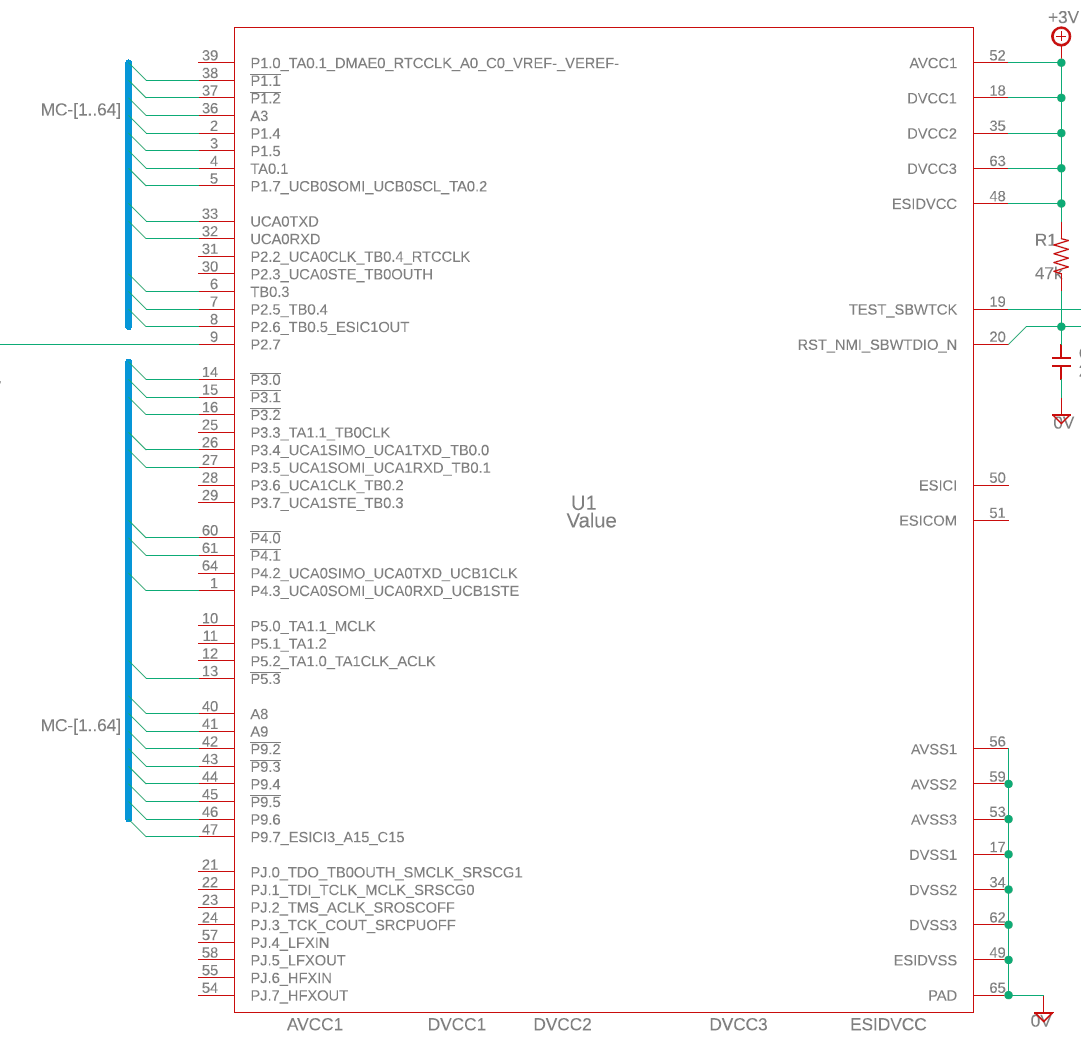


Figure 36: Microcontroller Schematic

### Battery Circuit

This subcircuit remained mostly the same. The only major modification was the voltage divider that converts battery voltage to readable ADC voltage. Appropriate resistance values were selected for R3 and R4 that converted the battery’s highest voltage (4.2V) to 2.5V which was the target voltage for the microcontroller’s internal ADC module. It was realized later in the firmware development process, that the ADC also worked off of Vcc, however at the time the voltage divider seemed like a necessary feature. Shown in Figure 37 below is the battery sub-circuit.

Battery Circuit

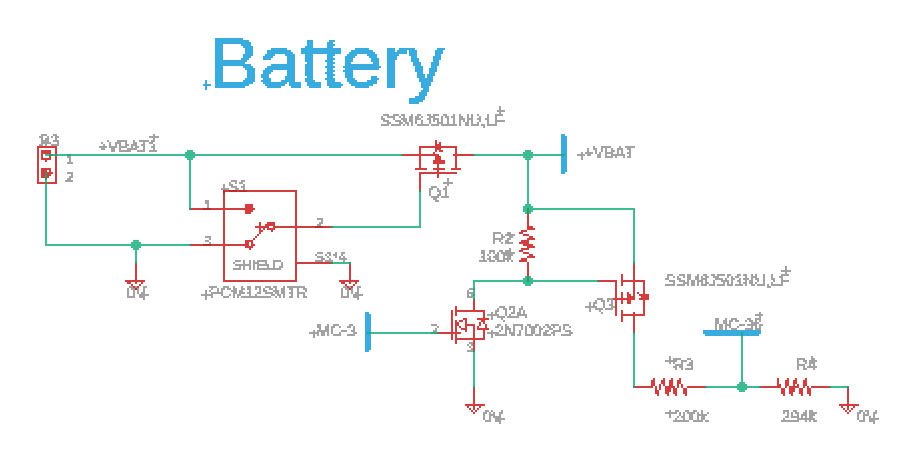


Figure 37: Battery Circuit

### Indicator LEDs

See Section 12.2 above for more details. Shown in Figure 38 below is the final indicator LEDs schematic for Rev. 1.

Indicator LEDs

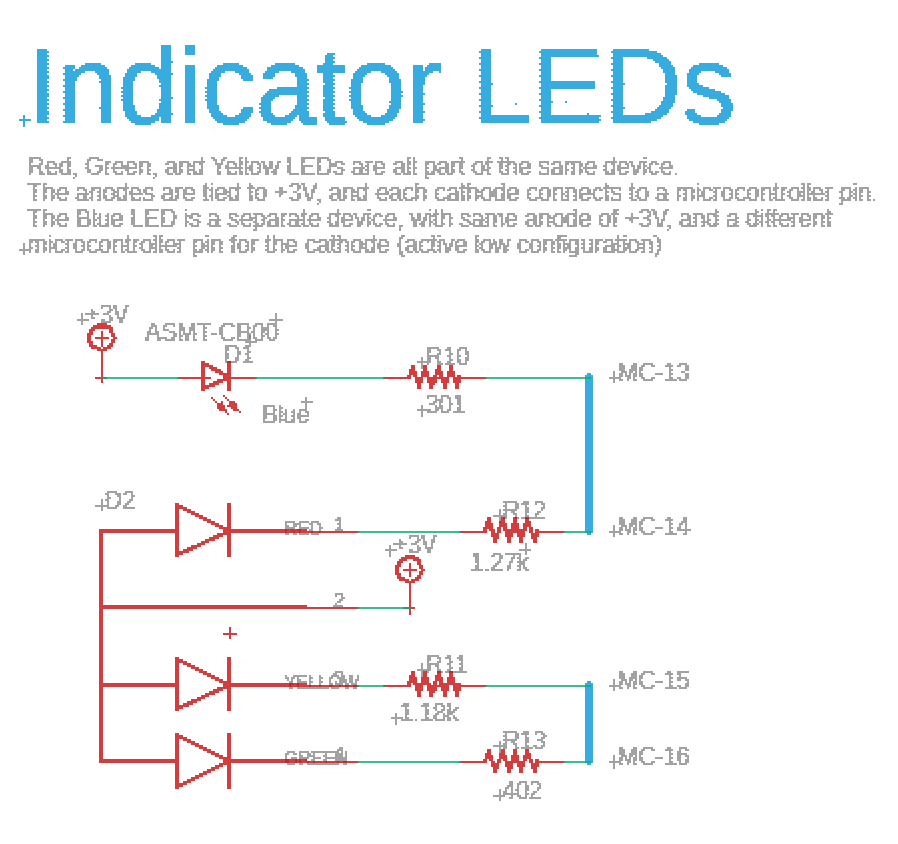


Figure 38: Indicator LEDs Schematic

### Push Buttons

Capacitors were added at the button GPIO pins so that this state cannot change as abruptly. This helped to improve the stability of the design. Shown in Figure 39 below is the push buttons circuit.

Push Buttons

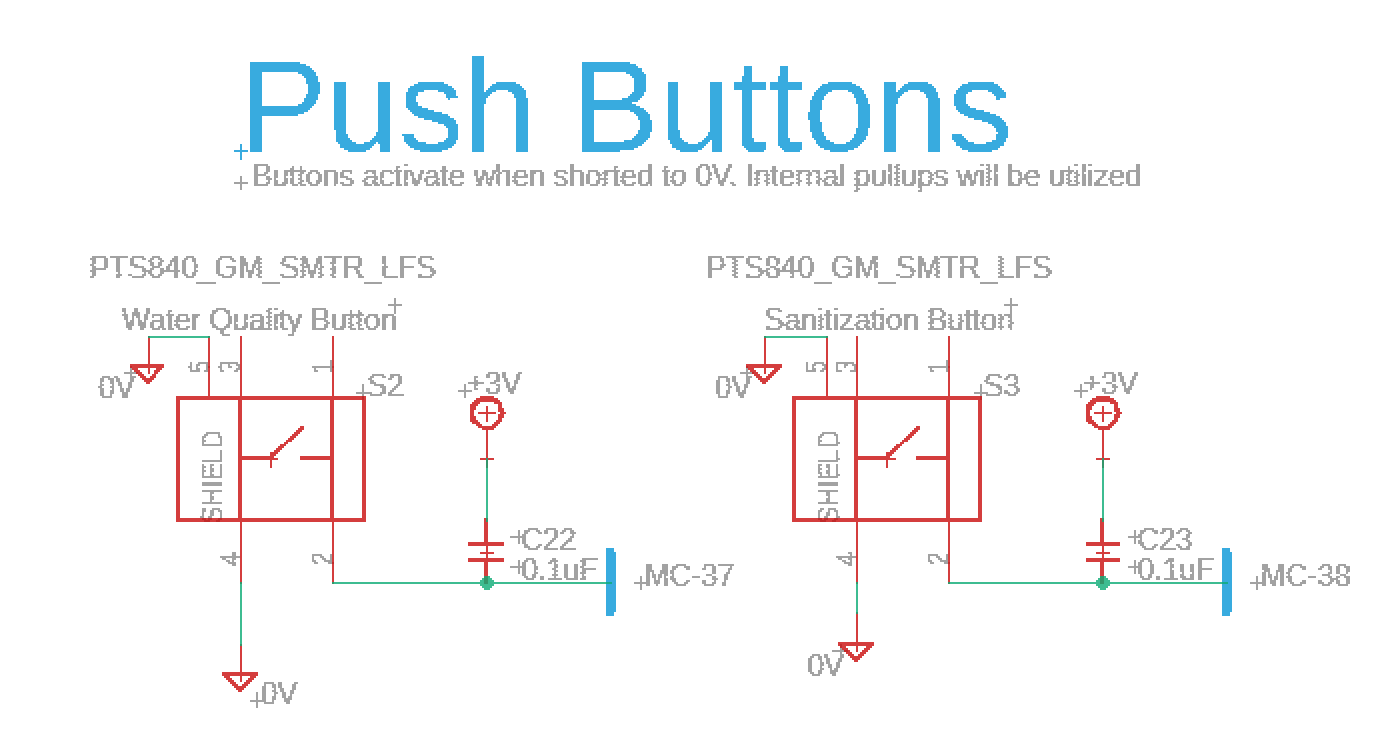


Figure 39: Push Buttons Schematic

### Laser Diode

The regulation mechanism for this circuit was the largest overall change. The previous current limiting diodes proved to be incompatible with design. The laser diode required large amounts of current which would result in using the current limiting diodes in parallel. However, this product was not intended for that use case. The solution would have involved implementing a transistor amplifier circuit to boost the current limiting diode’s capability. It was decided instead that resistors could be used to effectively regulate the laser diode current. Resistors were also placed at the gate of each MOSFET in case its behavior would need to be modified at a later time. A ‘virtual’ curve trace was created for the laser diode and used to determine the resistor’s size and value. Shown below in Table 42 is the laser diode curve-trace table.

Laser Diode Curve Trace Table

|  |  |  |  |
| --- | --- | --- | --- |
| Current | Voltage | Current | Voltage |
| 89 | 2.048 | 150 | 2.098 |
| 95 | 2.053 | 160 | 2.102 |
| 101 | 2.057 | 165 | 2.108 |
| 107 | 2.06 | 210 | 2.133 |
| 110 | 2.071 | 281 | 2.23 |
| 115 | 2.074 | 347 | 2.34 |
| 120 | 2.077 | 358 | 2.42 |
| 130 | 2.085 | 390 | 2.6 |
| 140 | 2.092 |  |  |

Table 42: Laser Diode Curve Trace Table

The previous table was used to project the forward voltage necessary across the laser diode for each desired power mode. From here, the effective resistance of the MOSFET had to be considered, and then the resistors could be selected. Note: the resistances shown for high power mode are for one of the two parallel resistors. For example, two 30.9 ohm resistors would result in a current of approximately 211 mA in high power mode. Using the measured voltages and currents, the resistor values and sizes (based on power consumption) were then determined. See Table 43 below for these calculations. Also, see Figure 40 below for the final Rev. 1 laser diode schematic

Laser Diode Resistor Calculations

|  |  |  |
| --- | --- | --- |
|  | Power Mode | |
|  | Low | High |
| Desired Current (mA) | 110 | 210 |
| Laser Diode Voltage (V) | 2.071 | 2.133 |
| Resulting Resistor Voltage (V) | 3.929 | 3.867 |
| Initial Resistance (Ω) | 35.7 | 36.8 |
| MOSFET Est. Resistance (Ω) | 2.27 | 2.86 |
| Adjusted Resistance (Ω) | 33.4 | 31.1 |
| EIA-Resistor value (Ω) | 33.2 | 30.9 |
| Estimated Final Current (mA) | 111 | 211.1961 |
| Total Resistor Power (W) | 0.43 | 0.812 |
|  |  |  |

Table 43: Laser Diode Resistor Calculations

Laser Diode

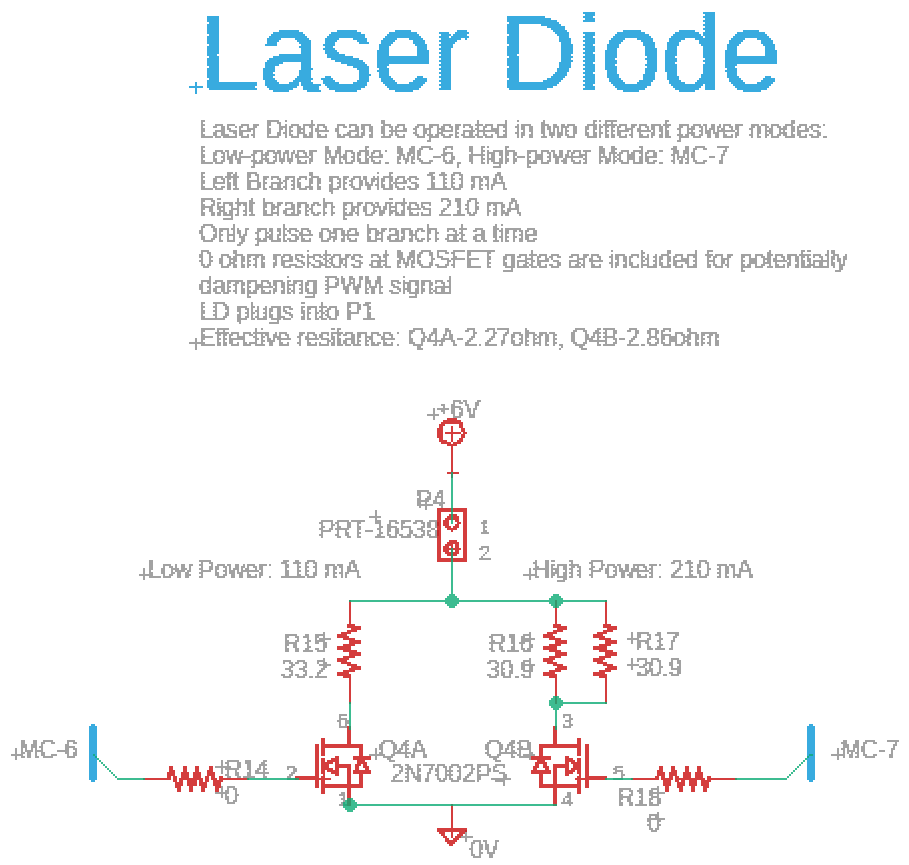


Figure 40: Laser Diode Schematic

### Analog Inputs

The photodiode circuit resistors were slightly modified for more deterministic behavior. Also, a voltage divider was included to convert 3V to 2.5V for the photoresistor ADC read. Shown below in Figure 41 is the analog inputs schematic.

Analog Inputs Schematic

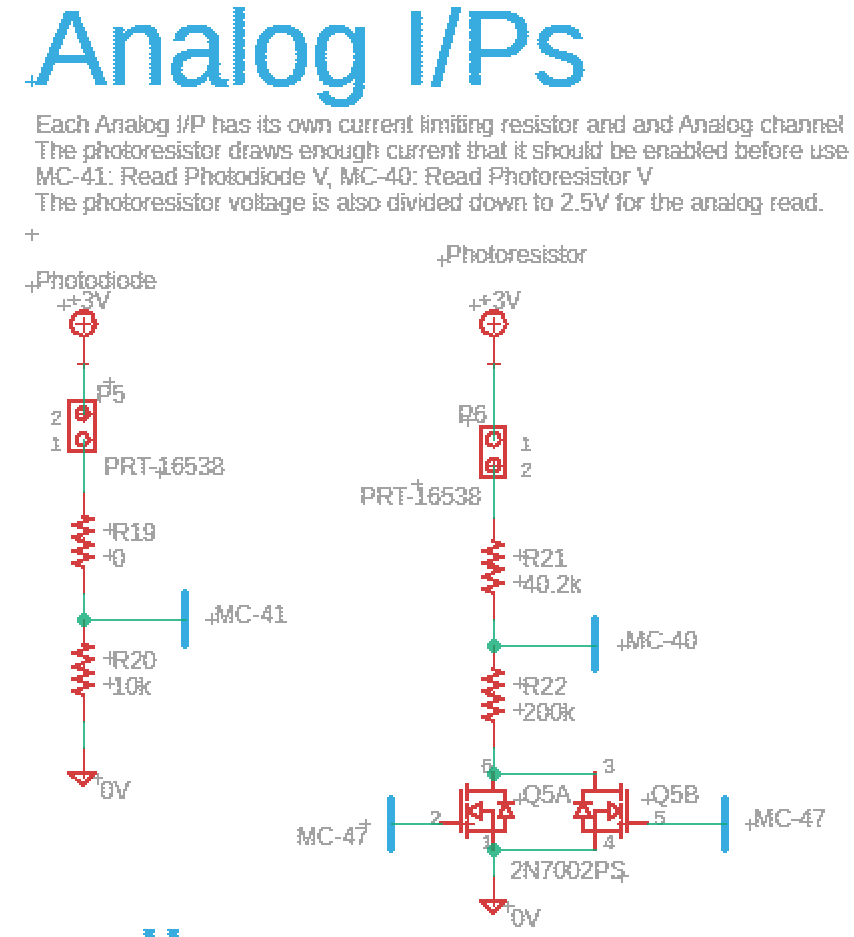


Figure 41:Analog Inputs Schematic

### UVCs and Motor

This circuit was slightly modified so that the board can either interface with a servo motor (previous implementation) or a stepper motor. The stepper motor required additional GPIO connections as well as a direct connection to 0V since it doesn’t have significant current draw. Shown below in Figure 42 is the schematic for the UVCs and the motor. The motor was operational using this design, but equipment failure on the motor itself (separate from this design) led to the abandonment of its usage.

UVCs and Motor Schematic

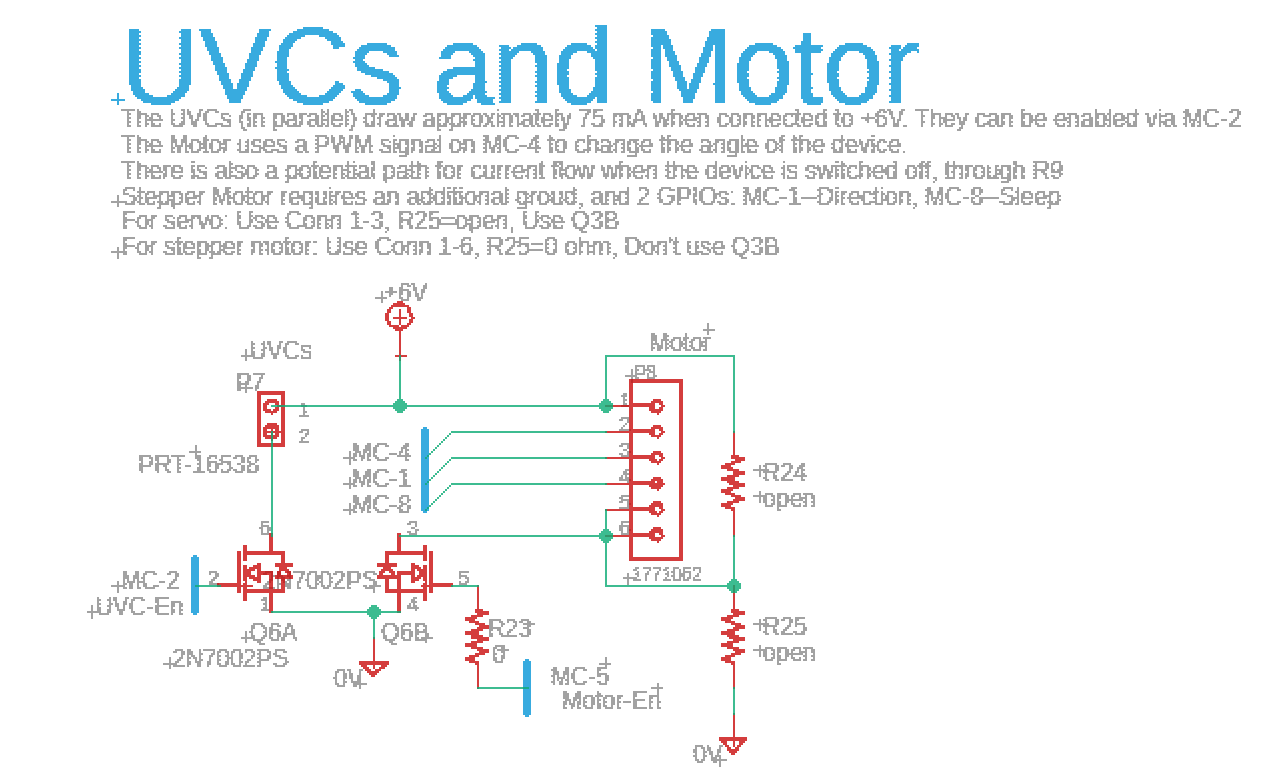


Figure 42: UVCs and Motor Schematic

### Bluetooth Module

A 0.1 uF bypass capacitor is now attached to the VDD pin. The rest of the Bluetooth module schematic follows the preliminary design. Shown in Figure 43 below is the Bluetooth module schematic.

Bluetooth Module Schematic

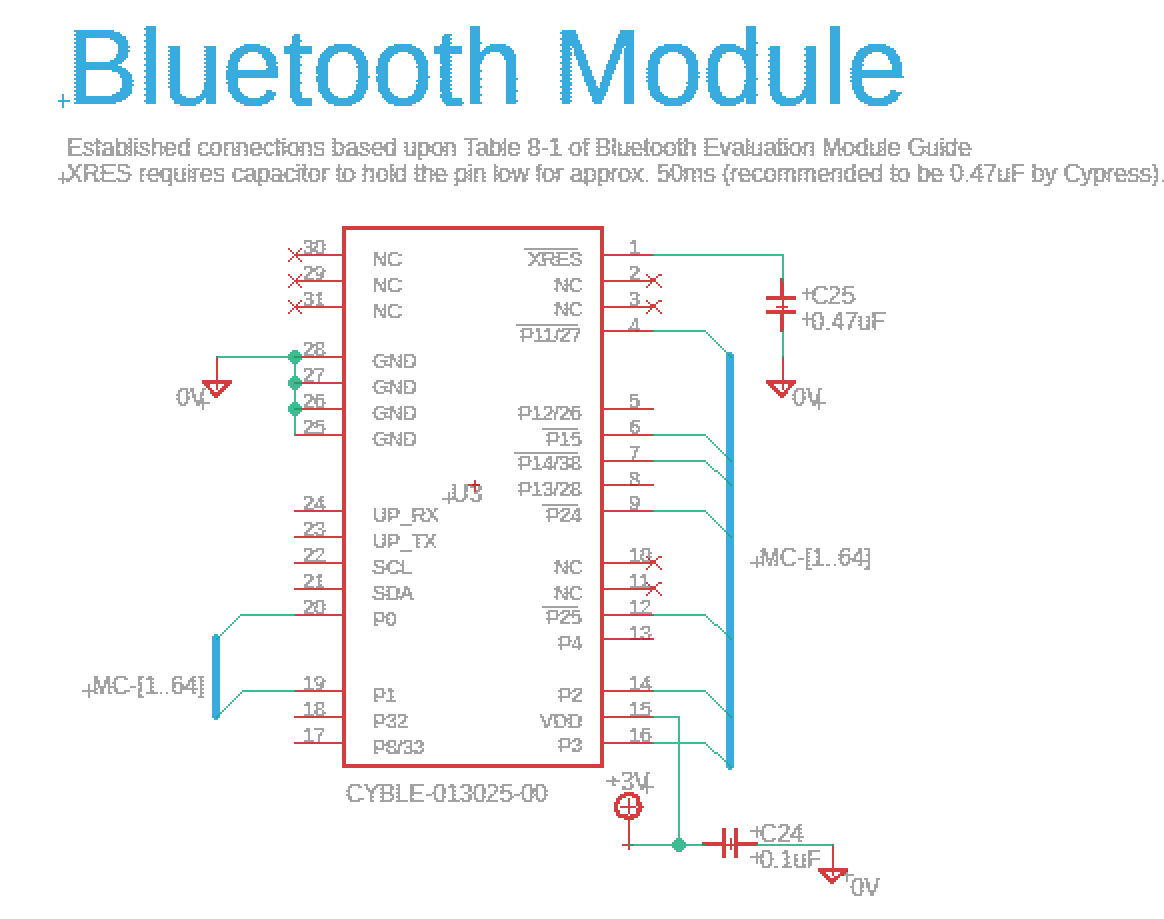


Figure 43: Bluetooth Module Schematic

### Reed Switch

The reed switch schematic remained the same as the preliminary design. Shown below in Figure 44 is the reed switch schematic.

Reed Switch Schematic

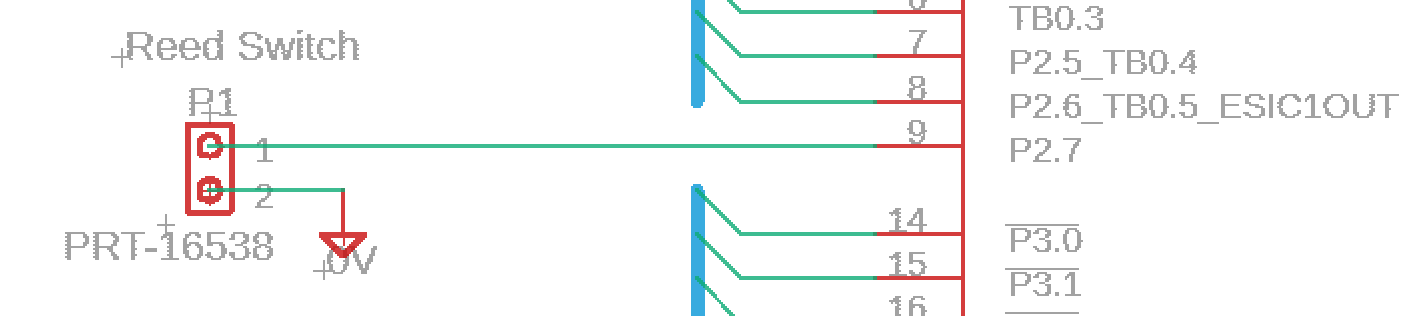


Figure 44: Reed Switch Schematic

### Programming Header

Once more information was known about how to flash the TI microcontroller, An eight-pin programming header was introduced into the design. This made it simple to attach wires and utilize the MSP430FR6989 Launchpad’s EZ-FET tool. A resistor and capacitor were included on the reset pin, per TI’s recommendations. Also, a back-channel UART was included to make the device easier to debug. Lastly, +6V and +3V were also included as test points throughout the prototype process. Shown in Figure 45 below is the programming header schematic.

Programming Header Schematic

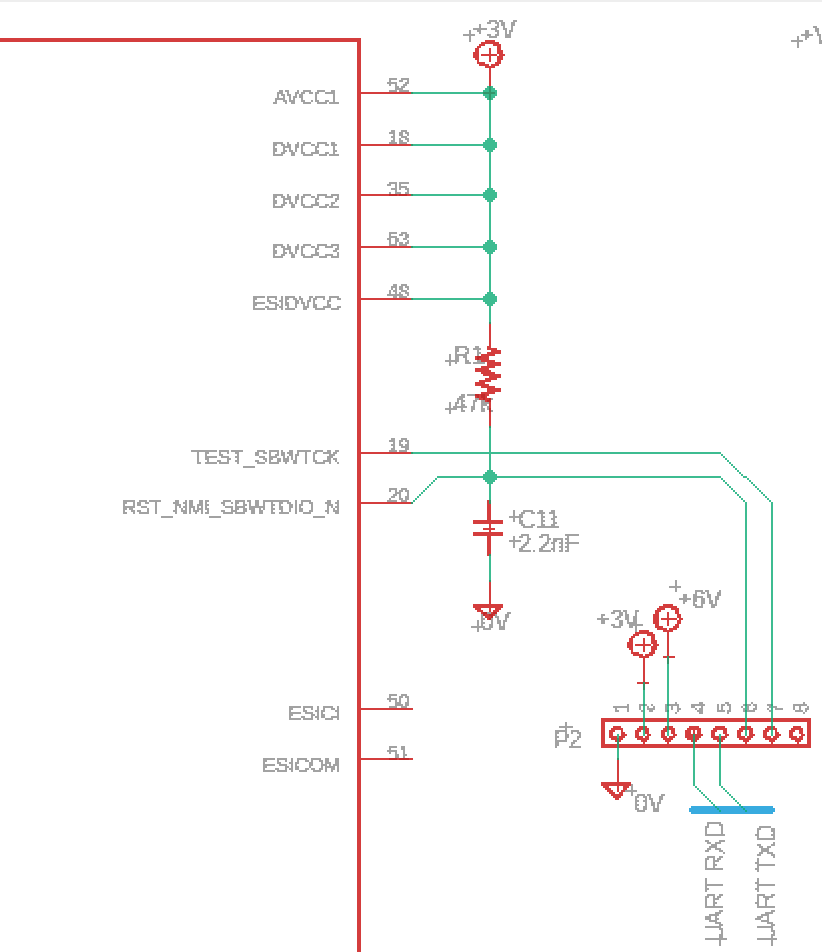


Figure 45: Programming Header Schematic

### Comprehensive Rev. 1 Schematic

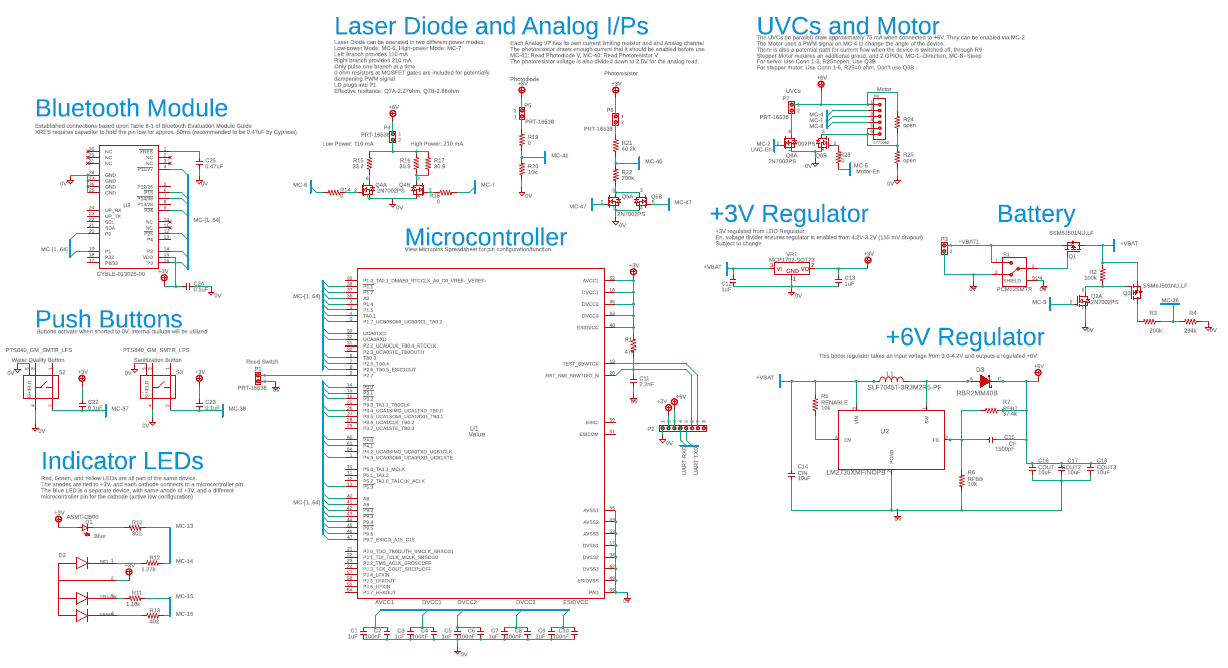


Figure 46: Comprehensive Rev. 1 Schematic

## Printed Circuit Board Design

Once the schematic was finished, it was time to design the printed circuit board. This process started with creating a board outline that met project requirements (radius of 2900 mil) and adding drilling holes for mounting the PCB in the 3D printed housing. Then the components were organized in order to minimize the amount of crossing unrouted traces. All external headers were placed on the bottom of the two-layer board, and the micrcontroller, Bluetooth module, and voltage regulators were placed on the top. The +3V regulator was designed using a heat sinking plane on the top and bottom to help improve the efficiency of the regulator. The +6V switching regulator sub-section or “island” was designed based on TI’s recommendations for minimizing noise for the IC. From here, all traces were routed by hand. This was a personal preference which helped to keep the board layout logical, coherent, and easy to follow. Busses of wires were utilized to again keep the board layout organized. This process took a significant amount of time and reiteration. But in the end, it yielded a very successful PCB and it was certainly worth the time and effort. Some of the other important attributes related to the PCB are the switches and LEDs for a user interface all at the bottom section of the top layer. Also, the Bluetooth module had a designated keep-out area for the antenna to work properly. It is worth mentioning that all design rule checks and electrical rule checks were performed prior to producing the final version of the PCB. Shown below in Figure 47 is the final PCB top view, whereas Figure 48 shows the bottom view.

Final Printed Circuit Board Layout--TOP

A map of a city

Description automatically generated with medium confidence

Figure 47: Final Printed Circuit Board Layout--TOP

Final Printed Circuit Board--BOTTOM

Graphical user interface

Description automatically generated

Figure 48: Final Printed Circuit Board—BOTTOM

## PCB and Component Assembly

Once the board was finished, all of the contract manufacturer files had to be generated. This involved extracting the GERBER files and centroid file from EAGLE. The GERBER files are used by the contract manufacturer to create the PCB, and the Centroid file is used to create origins for each component that needs to be placed on the PCB. At this time a Bill Of Materials (BOM) had to be created. This was a list of all required components that needed to be purchased and attached to the printed circuit board. For FLASC, it was decided that the boards were to be assembled by PCBWay. This was because group members already had experience working with them and did not want to choose an unfamiliar manufacturer for the project. Even though this option was more expensive (boards could have been ordered overseas, and then assembled locally in Lake Mary) this ended up being a great choice because the boards came out working effectively and looking very professional. It also is worth mentioning that prior to assembly, the pre-purchased microcontrollers had to be shipped to PCBWay in China. Shown below in Table 44 is the Bill of Materials that was given to the contract manufacturer for this project.

Bill Of Material



Table 44: Bill of Materials

Shown below in Figure 49 is the final printed circuit board for FLASC.

FLASC Printed Circuit Board

A picture containing text, electronics, hard disc

Description automatically generated

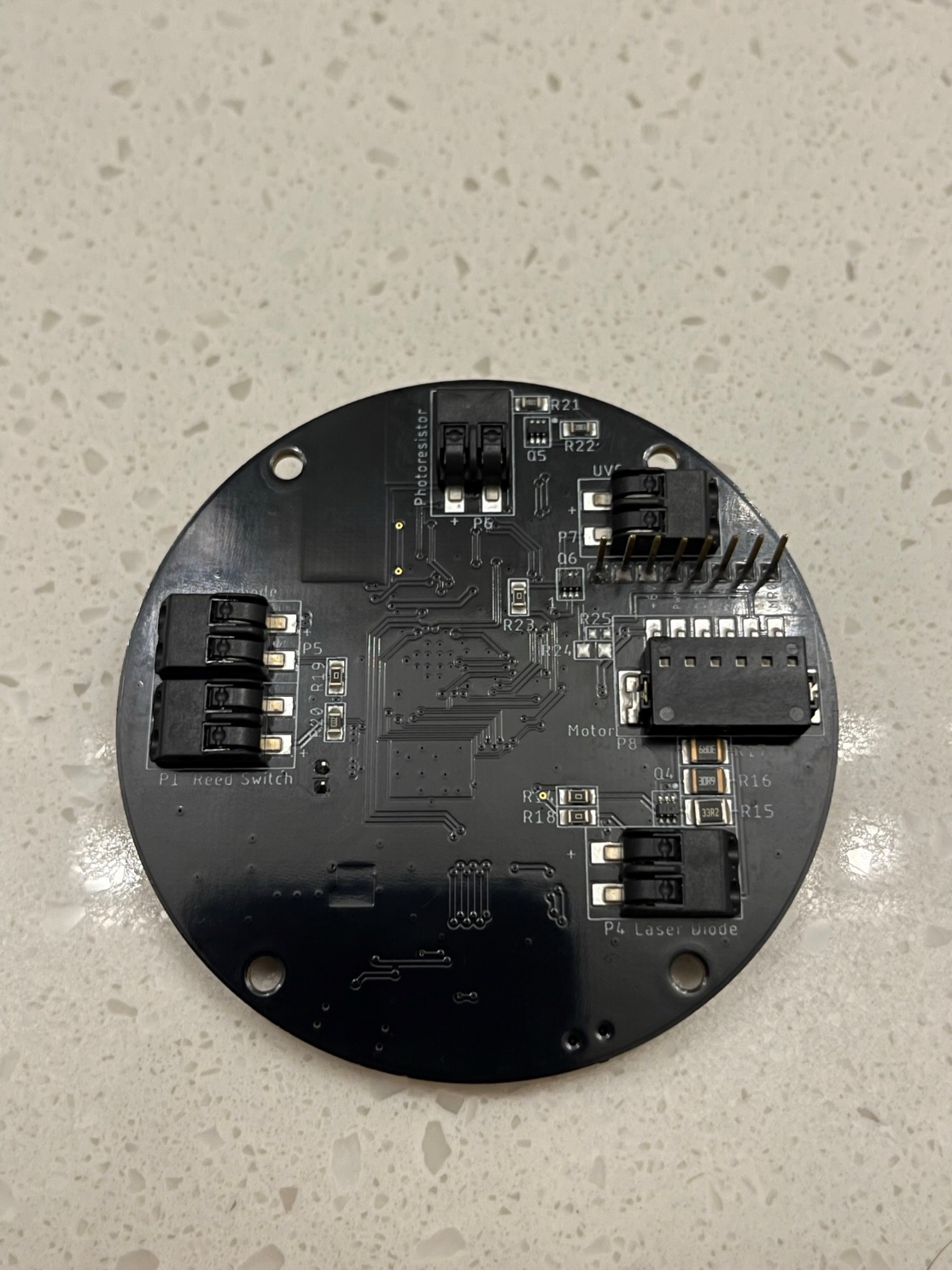


Figure 49: FLASC Printed Circuit Board

# Water Clarity System Design

## Spectrometer V2 Project Implementation

Originally, the water clarity system was intended to be compact, circular Raman spectrometer. The first version was never implemented as it was solely theoretical (utilizing a CMOS/CCD chip). Implementation of the second version of the spectrometer was less output intensive but required both output and input signals. The spectrometer module's servomotor required inputs to cycle through positions. The servomotor also required typical power supply connections (+6V and GND), and then utilized a PWM signal to change the angle of the device. This was easily accomplished through the microcontroller’s timing capability. From here, the PWM signal was driven out on a pin thwas configured to have PWM capability. The spectrometer module's output was an analog voltage across the module's photodiode- which, while much less complicated than a composite video signal, still required a voltmeter or equivalent system to convert a microvolt response from the diode into a value that represented said response within a reasonable (1 mV) degree of precision. Using a microcontroller with a built-in ADC would suffice for this job. It was important that it had enough resolution (within a small voltage range) to be able to detect such minute changes in voltage. More specifically, the output of the photodiode would serve as an analog input to the microcontroller, that was then converted to a digital signal and interpreted/analyzed for the purpose of water quality.

## Spectrometer V3 Project Implementation

The third implementation replaced the diffraction grating with a prism once again due to a change in the sanitizer design. As the sanitizer was updated to use LEDs instead of a laser, the spectrometer was forced to adopt its own stimulation source. Since the sample was a (relatively) large reservoir of water, the traditionally popular and widely available infrared diodes were off the table due to water's high absorption coefficient for that regime. Instead, a visible green laser was added as a pump source for the sample. A prism was interchanged with the previous diffraction grating due to a swap to visible ranges as transmission was no longer a significant concern in affordable optical glasses like NBK-7. The device was also updated to have a stepper motor instead of a servomotor. The stepper motor was attached to a break-out motor driver board from Pololu capable of powering the device, running with low sleep current, and micro-stepping the motor. The latter feature being especially helpful with the level of precision required for a water quality analysis system. This design utilized a pin for stepping through a rising edge set at a fixed frequency as well as pins for enabling the motor and changing the direction of the motor. Unfortunately, due to limitations in product availability, and a limited number of motors to test out, this version of the spectrometer had to be abandoned following equipment breakage compounded by shipping delays. The only stepper motor available fused, so the spectrometer had to be re-designated as a water clarity system.

## Water Clarity System Project Implementation

The final iteration of this system involved a laser diode and phototransistor for determining water clarity based on Boolean logic and a set ADC threshold. A visible green laser diode was enabled for two seconds, and then an ADC reading was taken on the series branch of the phototransistor. The threshold value for water clarity Boolean logic was set based on testing. The final system was able to distinguish distilled water from more opaque liquids such as coffee but also detected more subtle differences- specifically, a sample of water contaminated with 4 ppm Fluoride.

# Embedded Firmware Design

## Design Overview

After driverlib was added to the project. The first step to firmware design involved creating Initializer.c to establish all of the defines for components in the project, as well as initialize all pins to defined states at startup. This also involved configuring unused pins as pulled high inputs to avoid unnecessary current draw. From here, the team collaborated and came up with the basic tasks that the firmware needed to accomplish. Then the code was assembled iteratively, one building block at a time. This method allowed each block to be thoroughly tested for bugs and general functionality. As more blocks were added, more parameters had to be considered in regards to interdependence between code blocks. This process continued, alongside consistent uploads to the GitHub code repository, until more finalized versions of the codebase were created, tested, and implemented. It is important to note that this project was built on BareMetal (without an RTOS) so a blocking variable was utilized throughout the code that would only allow one process to execute at a time. This was done so that device modes could not interrupt each other, and because dynamic priority was unable to be assigned to the microcontroller. This section will outline the final states of primary functions that were utilized in the FLASC embedded firmware.

## Main.c

The main function began with initialization and then a battery read was performed. From here the code then entered the stand-by status loop. Every iteration of this loop would check to see if analyze, sanitize, or the exporter needed to be called. If nothing needed to be called, then the device entered low power mode.

The main function relied on interrupt service routines to handle the control logic for the program. So, whenever a button was pressed, an ISR was called and debouncing was performed to determine which button was pressed. If the sanitization button was pressed, a global StartSanitization Boolean variable was set and then Sanitize.c was called. If the analysis button was pressed, then a timer for 2 seconds was set to determine whether or not the button was being held down. If the button was held down, then the exporter variable was set to call the exporter. Otherwise, the analysis variable was set to trigger analysis. the safety device needed to have a rapid response, all interrupt service routines (and sub-functions) were kept as short as possible. This ultimately allowed Reed.c to “assume” control whenever the cap was removed. Interrupt service routines in standby mode really only set the variables necessary to activate other device modes.

Whenever the sanitization button was pressed, analysis and the exporter were called automatically at the conclusion of each prior event. This meant the user had the choice to either perform one full cycle (sanitization, analysis, and export), or call independent modes. So, logic also had to be included to account for various permutations that the user would perform. For example, the exporter only sends the water clarity when the sample is valid (a complete cycle of sanitization was run). So, if the exporter is called when the sample is not valid, then only the battery percentage is transmitted.

Whenever the hardware recognized an interrupt, the device would exit low power mode and run through the stand-by status loop to determine which mode needed to activate. From here, before every device mode (except for reed switch polling) was called, a battery read was performed. The aforementioned device modes would only execute if the battery life was above 20%. Anytime a device mode tried to activate but didn’t have enough battery life, a blinking red animation occurred. Because all ISRs, and sub-functions were kept as short as possible, the microcontroller spent the majority of its time in low power mode and asleep. Shown below in Figure 50 is the flowchart for Main.c.

Main.c Flowchart

Diagram

Description automatically generated

Figure 50: Main.c Flowchart

## Reed.c

From the start of this project, it was always a critical requirement for the safety device to be very reliable and responsive. So, whenever a falling edge was observed on the reed switch pin, the code immediately entered reed switch polling mode. Inside of the ISR for the reed switch, the UVCs, green laser, and all button interrupts were disabled Also, the sample was deemed “no longer valid”. This ensured that the safety feature would respond nearly instantaneously. This also meant that if the cap was removed, then the water quality could no longer be guaranteed, so sanitization would have to be performed again. From here, Reed.c entered reed switch polling mode. This pin had to be polled every second because a shorted reed switch created a constant current draw of 100 µA. So, every second the reed switch microcontroller pin was checked to see if it was low. If it was low, this pin was then configured as a pulled low input, and a timer was set for one second while the micrcontroller was asleep. After a second then the pin was configured as a pulled high input, and once again the pin was checked. This process continued until the pin was high during the check--which indicated that the cap was re-attached to the bottle. At this point the blocking term was released, and the device entered back into the standby status loop and resumed normal operations. This feature certainly could have been refined but based on the time constraint of the project it seemed like a suitable solution. However, it required the user to be strongly advised not to leave FLASC in reed switch polling (or safety) mode for a long period of time, since it would unnecessarily drain the battery. Shown below in Figure 51 is the flowchart for Reed.c.

Reed.c Flowchart

Diagram

Description automatically generated

Figure 51: Reed.c Flowchart

## Sanitize.c

Sanitize.c began by ensuring that the cap was on the bottle, and then starting sanitization as long as the previous condition was true. A yellow LED was turned on to indicate that sanitization was taking place, and a ten second timer was set while the microcontroller went to sleep. After ten seconds, the UVC feedback system kicked in. The photoresistor was enabled, and (after a brief moment) the ADC reading was taken. Then conditional logic checked to see how the measurement related to the threshold voltage. If the measurement was less than or equal to the experimentally determined threshold then the UVCs were turned off, sanitization canceled, and a blinking red error animation occurred. Otherwise, a brief green blink appeared and sanitization continued for the remainder of the three minutes, while holding the blocking term and preventing other device modes (except for the safety mode) from activating. At the end of the three minutes, the yellow LED turned off, and the UVCs were disabled. Then, the analyzer was called. It is worth noting that the majority of this logic occurred in brief ISRs using conditional logic, so that the sanitizer wouldn’t hold system resources for a long period of time. In other words, virtually immediately after the cap is removed during sanitization, FLASC enters safety mode. This logic also holds true for the analyzer, and exporter. Shown in Figure 52 is the flowchart for Sanitize.c.

Sanitize.c Flowchart

Diagram

Description automatically generated

Figure 52: Sanitize.c Flowchart

## Analyze.c

The analyzer functions in a very similar fashion to the sanitizer, except that it only runs for about two seconds. Whenever analysis starts, it turns on the blue LED, as well as both branches of the laser diode to run it in high power mode. A timer is set for two seconds (while sleep occurs) and then an ADC reading is performed on the photodiode. Boolean logic is then used (based on experimentation) to either blink green or red 10x to indicate clear water clarity or unclear water clarity, respectively. After the analysis is finished, the laser diode and blue LED are turned off, and the exporter is automatically called. Shown below in Figure 53 is the flowchart for Analyzer.c.

Analyzer.c Flowchart

Diagram

Description automatically generated

Figure 53: Analyzer.c Flowchart

## Export.c

The exporter takes in three parameters: the result of the analyzer (sample), the battery percentage, and ValidSample. ValidSample is the Boolean variable used to ensure that water clarity is only transmitted if a sanitization has been completely run. The exporter uses standard UART functions to transmit characters to the Bluetooth module that automatically transmit to the CySmart app. If the sample is valid, then the battery percentage, and water clarity are transmitted to the app. Otherwise, only the battery percentage is transmitted. The exporter is called automatically after analysis. But if the user didn’t have their phone connected they can always connect and export at a more convenient time as long as they do not remove the cap or run another cycle of sanitization. To signify the wireless export, a blue light blinks five times. Then the exporter is finished and the device returns back to its standby operating mode.

## BatteryRead.c

Whenever a battery read is performed, the battery read circuit is enabled, and a timer is set to allow the circuit to settle. Then an ADC reading is performed. From here the 12-bit result is converted into an integer percentage based on a linear interpolation of desired battery life constraints. For this curve fit, 4.2V was designated 100%, and 3.4V was designated 0%. Even though this technically is not the end-of-life (EOL) voltage. This limit was artificially established to ensure the device runs sanitization at peak performance capability. It is also important to note that a linear interpolation is not the best application for a lithium polymer battery. LiPo batteries typically maintain the same voltage for the majority of a cycle, and then start to decay rapidly as they approach the EOL voltage. However, since this project didn’t contain any CpE members, this seemed like a suitable application.

# Prototyping and Breadboards

## UVC Sanitization

A basic breadboard prototype was constructed (see Figure 54: Sanitizer Optical Setup below) and was successfully used to sterilize water from the UCF reflection pond for a demonstration. These LEDs proved too large for realistic usage in the FLASC, but the principle and voltages remained unchanged. Shown below in Figure 54 was the optical set-up used for this test.

Sanitizer Optical Setup

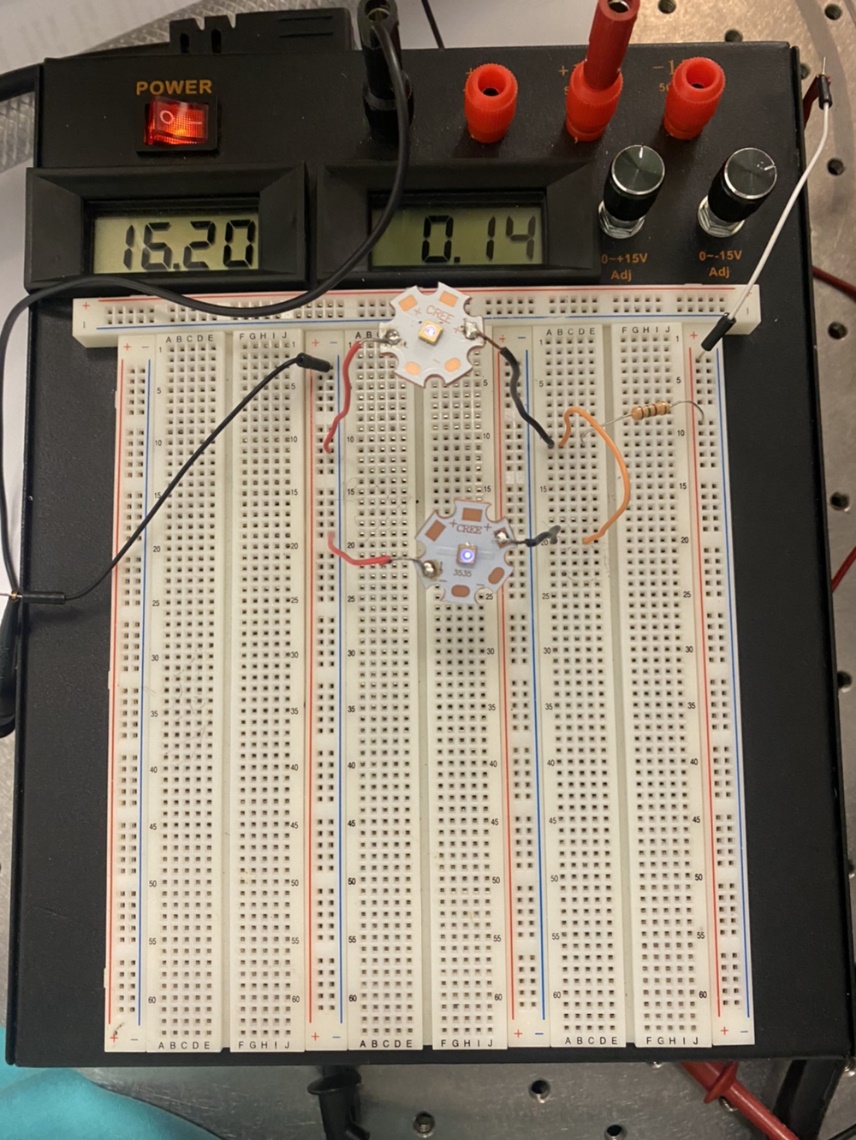


Figure 54: Sanitizer Optical Setup

## Bluetooth Module

In order to test the microcontroller code that needed to communicate with the BLE module, it was very helpful to utilize the CYBLE-013025 Evaluation Board in conjunction with the MSP430 LaunchPad for connectivity purposes. The goal was to have a development environment in which the connections made on the final printed circuit board could be simulated ahead of time to ensure project functionality—this is listed below in Table 45. These port and pin numbers were printed on the evaluation boards that were obtained and each pin was conveniently accessible via the development board header pins. Therefore, it was necessary to connect the appropriate pins with the use of jumper cables. The resulting prototype configuration for BLE testing can be seen below in Figure 55.

CYBLE-013025 Pin Connections

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Assignment** | **CYBLE Pad Number** |  | **MSP430 Pin Number** | **MSP430 Function** |
| UART\_RX | P2 | 14 | <--------------> | 33 | UCA0TXD |
| UART\_TX | P0 | 20 | <--------------> | 32 | UCA0RXD |
| UART\_RTS | P1 | 19 | <--------------> | 46 | P9.6 |
| UART\_CTS | P3 | 16 | <--------------> | 44 | P9.4 |
| CONNECTION | P14 | 7 | <--------------> | 61 | P4.1 |
| CYSPP | P27 | 4 | <--------------> | 60 | P4.0 |
| DATA\_READY | P15 | 6 | <--------------> | 43 | P9.3 |
| LP\_MODE | P24 | 9 | <--------------> | 42 | P9.2 |
| LP\_STATUS | P25 | 12 | <--------------> | 45 | P9.5 |
|  |  |  |  |  |  |
| VDD | VDD | 15 | <--------------> | VDD |  |
| GROUND | GND | 25 | <--------------> | GND |  |
| GROUND | GND | 26 | <--------------> | GND |  |
| GROUND | GND | 27 | <--------------> | GND |  |
| GROUND | GND | 28 | <--------------> | GND |  |

Table 45: CYBLE-013025 Pin Connections

EZ-BLE Eval Board Connections

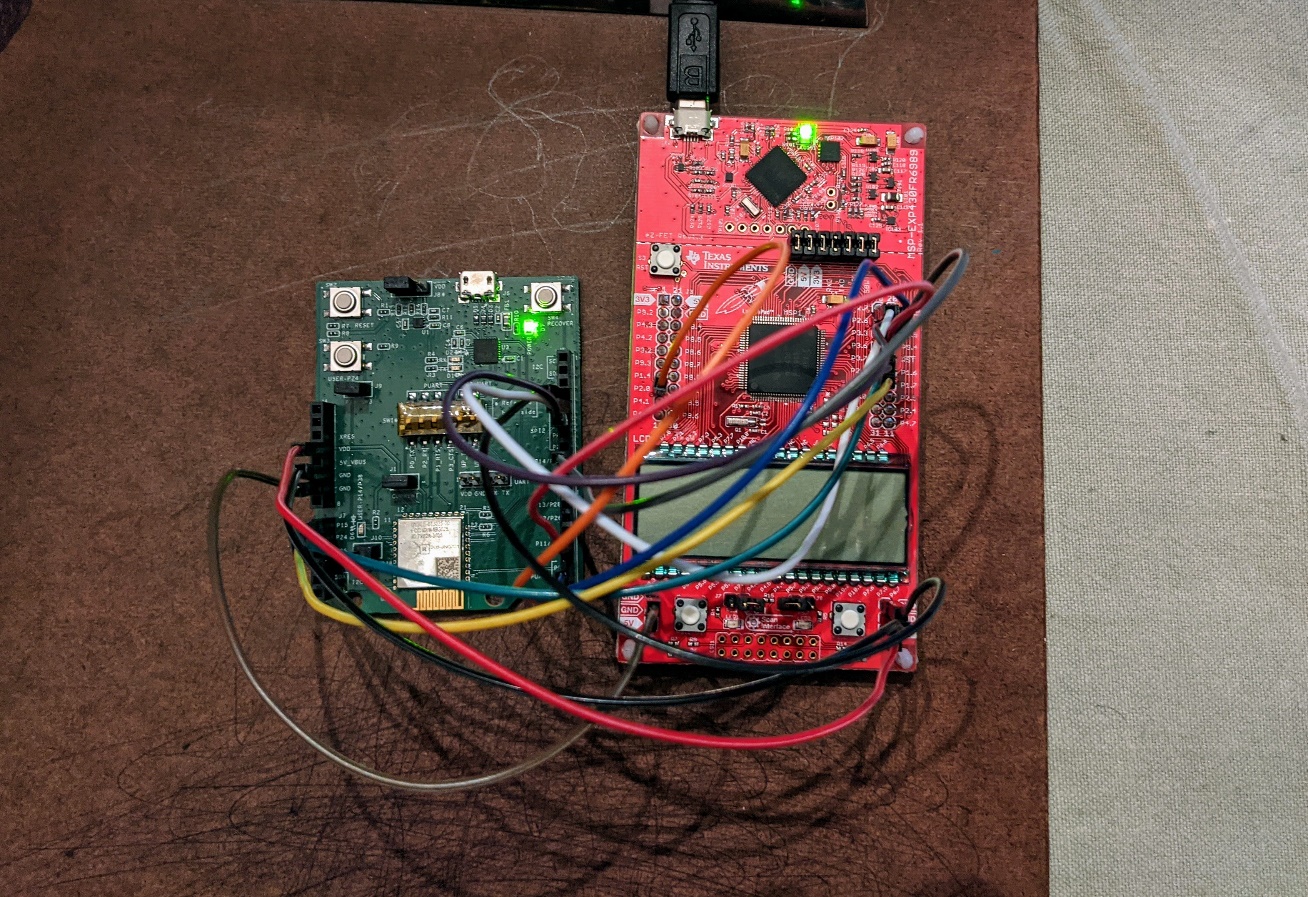


Figure 55: EZ-BLE Eval Board Connections

The final considerations for the development environment that needed to be dealt with before the coding process were the DIP switches and jumpers on the pre-made boards. Jumper J8 on the Cypress board was used for selecting the VDD voltage level, which needed to be supplied to the board via a 3.3V header pin on the MSP430 LaunchPad. There was also jumper J10 on the Cypress board that was initially connecting P14/P38 to an LED on the evaluation board, but this needed to be adjusted because P14 was later on used as the UART receiver input for the CYBLE-013025 BLE module. All 6 DIP switches on the Cypress board needed to be set to the OFF position in order to put the board into “General Application Mode”, which bypasses the use of the micro-USB connection so that we can access each port via the female headers and simulate the direct connection that was intended between the two integrated circuits. All of the jumper and DIP switch settings can be viewed in the evaluation board documentation (18).

The CYBLE modules on the final FLASC printed circuit board already came flashed with the aforementioned EZ-Serial WICED BLE Firmware Platform, so we were able to immediately test the functionality of the BLE module in CYSPP mode as we intended to use in the final product. However, in order to ensure that we were aware of what settings the BLE module currently had programmed on it, the simplest option available was to connect the Cypress evaluation board to a computer terminal via USB, put the module into command mode, and send some pre-written commands over UART in order to interact with the EZ-Serial API. This returned some binary (hexadecimal) messages back to the PC UART terminal of choice and indicated the current settings selected in the BLE module firmware. Originally, we thought the responses might not directly line up with the examples listed in the EZ-Serial WICED BLE Firmware Platform User Guide (11), and that we would have to either start figuring out how to use the Python script for converting the binary messages into text and/or download the EZ-Serial firmware from the Cypress website and flash the module that was currently embedded in the evaluation board. Fortunately, this was not necessary, and all messages were discernible right out of the box. Shown below in Figure 56 is an example of Bluetooth transmission obtained later on in product development observed using RealTerm serial console.

RealTerm Bluetooth Transmission Example

A picture containing graphical user interface

Description automatically generated

Figure 56: RealTerm Bluetooth Transmission Example

## Analyzer Prototyping

### Water Quality System v2 (Obsolete)

The water quality system's electronic components are visible in a prototype setup below with the laser (whose bulk was primarily a removable housing with lens which served as a heat sink and focusing system) visible on the upper right, servo and mirror on the upper middle, and photodiode on the lower left (in an open-circuit). The role of the laser driver was performed by a buck converter in this prototype, dropping a 5-volt output from the board to an appropriate voltage for the desired power level. The servomotor was connected directly to a 5-volt output from the board in this prototype as the servo had a dedicated, internal PCB and could handle a range of input voltages. The only control pin used in this prototype was P1.6, which was the PWM signal for the servomotor. This PWM signal was altered in the prototype by pressing buttons on the launchpad, which increased or decreased the duty cycle by an increment defined in the project code (needed to be greater than the dead bandwidth). Not shown is the ADC connection across the photodiode's series resistor which would eventually read the induced voltage, as this role was performed by a digital multimeter for the prototype. Shown below in Figure 57 is the prototype for this water quality system.

Water Quality Sensor Board Connections (Obsolete)

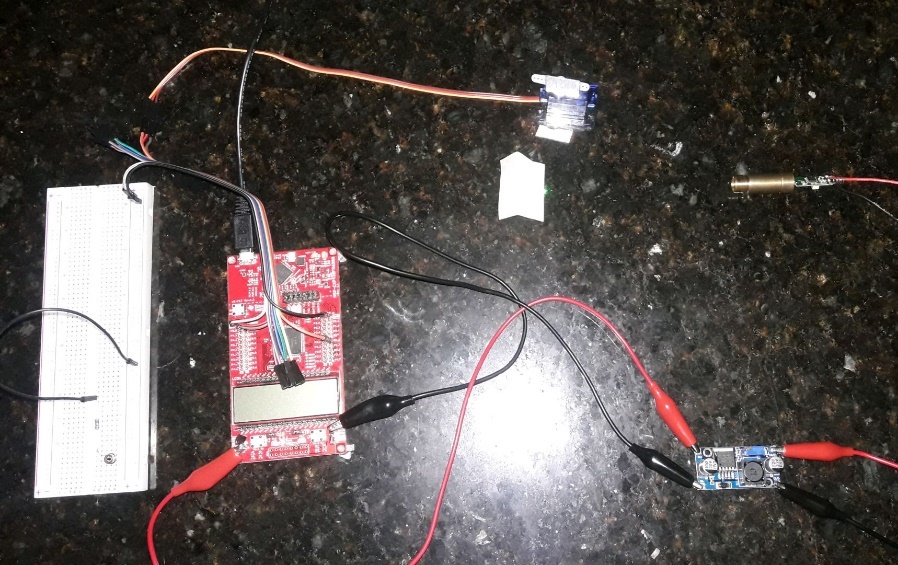


Figure 57: Water Quality Sensor Board Prototype

Below in Figure 58 is an incomplete prototype of the optical level which was created to house the spectrometer. Light from the viewing port on the surface-interface layer was levelled with its center approximately 5 mm above the optical level's surface by an adjustable mirror which was held in place using a small magnet. The light would then pass through a vertical slit before separating into a spectrum on the prism (not shown). Ray tracing was performed in FreeCAD using construction lines.

Optical Level (Obsolete)

A picture containing black, indoor, helmet

Description automatically generated

Figure 58: Optical Level (Obsolete)

### Water Clarity Analyzer

The water clarity analyzer was prototyped directly in the housing, as the system was simple enough to not require additional prototyping and the redesign was made after the housing was designed. This housing design can be found below in Figure 59. This led to immediate testing and results which were used to calibrate the analyzer.

Water Clarity Analyzer Prototype Structure

Diagram

Description automatically generated

Figure 59: Water clarity analyzer prototype layout.

# Device Operation

While our device was designed to operate in a fairly straightforward manner, some rules of operation will be established in the following sections to ensure that the device performs as intended.

## Sample Preparation

The user should select a source which is not, from a rudimentary firsthand analysis, dirty. Muddy, algae-infested, and obviously discolored samples should be avoided. The user should fill the bottle with less than 17 ounces of liquid- while the lower limit is less exact, the bottle should be filled at least halfway (8.5 oz) to ensure that the analyzer has enough sample to verify water clarity.

## General Device Operation

Once the sample is prepared, the user is now ready to utilize the FLASC for sanitization and analysis. Begin by turning on the device using the left-most slider switch (see the location indicated by the tweezers in Figure 60 in the left image). A red blinking light (reed switch polling mode) will be visible (see Figure 60 on the right image, where a red LED is visible) until the user places FLASC on top of the bottle and lines up the reed switch to the magnet on the bottle. Ensure that the red blinking light goes away before activating any standard device mode. A standard use case begins by pressing the sanitization button; however, the user could also press the analyze button to begin analysis or hold down the analyze button to export the data. It is also worth mentioning that if the battery is below 20% capacity, then the user will see five red blinks when attempting to start a device mode.

FLASC Power Switch and Indicator Light Locations

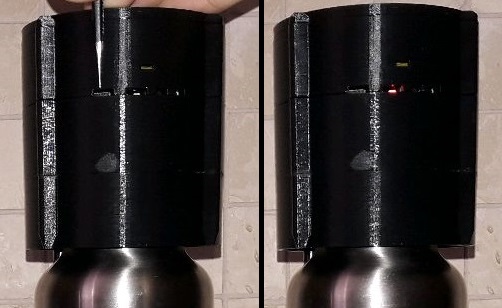


Figure 60: FLASC power switch (left) and LED indicator (right) locations.

This example will now outline the standard use case: The user should place the bottle on a flat surface and then press the sanitize button (seethe left image in Figure 61 for the location). A yellow LED will come on, and then after 10 seconds, the UVC feedback system will activate. If the UVCs are not working properly, then the device will blink the red LED five times and then terminate sanitization. If this error occurs, then the UVCs of the unit need to be replaced.

Assuming that sanitization passes its UVC check, sanitization continues for a total time of three minutes. Once it finishes, the water clarity analyzer is automatically called. At this point the yellow LED is replaced by a blue LED. Analysis takes approximately two seconds, and then the clarity result is shown. FLASC will either blink red or green ten times depending on the analysis result. For example, ten green blinks indicates that the water passed the clarity test. From here, the device will blink blue five times as it tries to export the data. Analysis can also be initiated directly using the analyzer button shown in the image on the right in Figure 61.

Sanitizer and Analyzer Buttons

A picture containing indoor, black, blender

Description automatically generated

Figure 61: Sanitizer (left) and analyzer (right) buttons.

For Bluetooth functionality, the user can decide if they want to connect their smartphone before the exporter is automatically called, or if they want to perform at a later and more convenient time. As long as the cap is not removed or FLASC is turned off, the user can export the results of the most recent sanitization at any time. To use the CySmart app, begin by ensuring that the FLASC is turned on and selecting a BLE device named “EZ-Serial”. Next, select the Gatt Database and the service. Subscribe to the “Write and Notify” characteristic and select indicate. Now, any data that FLASC tries to export will appear in the box titled “ASCII”. When looking at exported data it is important to mention that the only time the water quality will be sent is if sanitization (which also calls the analyzer). That is, if the cap is removed or if only the analyzer is called, the water quality will not be sent to the user’s smartphone.

## Using the App

The CySmart app is available from the app store on both Android and iOS. Once downloaded, the app prompts the user to turn Bluetooth connectivity on. The app then automatically starts scanning for Bluetooth Low Energy devices and shows a list of the device names along with their MAC address. The user should scroll through the list and select the device named “EZ-Serial” in order to bond with the FLASC. The CySmart app will present two different profiles that are available from the EZ-Serial firmware, the one that will allow us to establish the data pipe is named “Gatt DB”. There is only one service within the GATT database which is shown as “unknown service”; the user should select this option to proceed.

Finally, the user will see the three characteristics available for data transmission with the device. For easiest use, one may simply select either of the first two characteristics that are labeled as having a “write” capability. Specifically, the two options for data transfer are the acknowledged data characteristic (UUID: 65333333-A115-11E2-9E9A-0800200CA101) and the unacknowledged data characteristic (UUID: 65333333-A115-11E2-9E9A-0800200CA102). Selecting either of these two and then pressing the “Indicate” (for the acknowledged) or “Notify” (for the unacknowledged) will complete the process for establishing the data pipe between the smartphone and FLASC. Whenever a transmission is sent from the FLASC, the 20-byte payload will be shown on the app interface in both ASCII and hexadecimal form. The ASCII interpretation will show the current battery percentage of the FLASC as well as the status of the water if it has been sanitized since the last time the cap was put into place.

## Charging

The Qi standard for wireless charging is completely automated and does not require any special setup. Once any Qi compliant charging pad has been plugged in to a power source, the user may simply remove the FLASC bottle cap and place it upside down on the charging pad. A red LED will begin blinking within the FLASC to indicate that charging has begun (see Figure 62 below). In order to easily differentiate the red charging LED from the red LED blinking to indicate that the cap has been removed, the user may turn the device off from the power switch. This will also eliminate any unnecessary power expenditure while the device is charging.

Wireless Charging Orientation

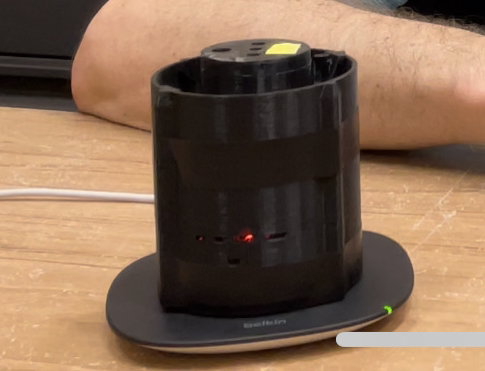


Figure 62: Wireless Charging Orientation

Once the battery has reached a full charge, the FLASC will stop the blinking red light and show a continuous green light. It is also worth mentioning that some Qi standard charging pads on the market have their own LED indicator to help the user ensure that the best alignment has been made between the transmitter and receiver. If this is the case, the user should slightly move the position of the FLASC until the Qi charging pad shows a green light. In general, the user may use the inscribed device name “FLASC” as a guiding mark to try to center the unit on the pad for the best possible charging efficiency.

## Recommendations for Prolonged Lifetime

In order to conserve the battery life of FLASC, it is highly recommended that the user turns the device off when it is removed from the bottle from an extended period of time. Reed switch polling mode will create a minute but continuous current draw that can unnecessarily use up battery capacity.

Extra care should be taken when handling the FLASC. Since the optical components inside are fragile, try not to drop the device at any time. And, whenever either sanitization or analysis is running, the bottle (and FLASC) should remain stationary on a flat surface.

# Testing

Components, systems, and programs required testing to verify their specifications and effectivity and to gather data regarding operational parameters which impact other systems.

## Components

Components are singular devices which were purchased from manufacturers and retailers without assembly by the project team. The tests that were run and their results are collated below in Table 46 and **Error! Reference source not found.** for the spectrometer and sanitizer respectively.

Analyzer Component Tests

|  |  |
| --- | --- |
| Component | Tests |
| Results |
| Servomotor (Obsolete) | Test 1: The PWM signal for the servomotor demo code needs to be verified on an oscilloscope. It is hard to ensure that the servomotor is properly configured without observing the PWM signal’s period and duty cycle on test equipment. |
| Result 1: Oscilloscope waveforms clearly show that the PWM demo code starts out with a 20 ms period and 1.5ms duty cycle. Then, then the duty cycle increases in 10 microsecond increments up to 1.506 milliseconds. |
| Stepper Motor (Obsolete) | Test 1: The stepper motor must be assembled with its driver board, a 47-microfarad capacitor across the motor power, and all connections required for operation in accordance with the manufacturer's wiring diagrams.  Test 2: The stepper motor code must be tested, and the stepper motor verified to be functional. |
| Result 1: Following the destruction of one driver board due to excessive resoldering, a spare driver board (which had been ordered concurrently to the first board) was successfully utilized.  Result 2: The stepper motor moved as directed, but after some testing the motor fused in place and no longer functioned despite a lack of apparent cause. |
| Optical Layer (Obsolete) | Test 1: The optical layer must direct a collimated (or relatively tight) beam from the surface-interface layer onto a direction on the optical layer. |
| Result 1: The adjustable mirror successfully directed the beam on the desired track. |
| Laser | Test 1: The laser's IV characteristics need experimentally observed. |
| Result 1: The laser's IV characteristics were plotted using a series ammeter and parallel voltmeter. The laser module begins lasing at 89 mA and 2.048 V, and was found to require ~2.07 V on the low setting (110 mA) and ~2.17 V on the high setting (220 mA). |
| Phototransistor | Test 1: The phototransistor must be placed in its designated position in the base of the microcontroller layer and the ADC values obtained for clean water and coffee. |
| Result 1: The ADC values for clean water did not drop below 300 while for coffee did not exceed 10. |

Table 46: Analyzer Component Tests

A curve trace was created for the laser diode, so that its I-V characteristics could be determined. Shown in Table 47 below is the resulting data from the test.

Laser Diode Electrical Characteristics Table

|  |  |
| --- | --- |
| Current | Voltage |
| 89 | 2.048 |
| 95 | 2.053 |
| 101 | 2.057 |
| 107 | 2.06 |
| 110 | 2.071 |
| 115 | 2.074 |
| 120 | 2.077 |
| 130 | 2.085 |
| 140 | 2.092 |
| 150 | 2.098 |
| 160 | 2.102 |
| 165 | 2.108 |
| 206 | 2.17 |
| 210 | 2.133 |

Table 47: Laser Diode Electrical Characteristics

Shown in Table 48: Sanitizer Component Tests are the tests and results run on the sanitizer components, and the UVC's output spectrum can be found in Figure 63.

Sanitizer Component Tests

|  |  |
| --- | --- |
| Component | Tests |
| Results |
| LED | Test 1: Check the optical spectrum of the LEDs to see what the actual peak wavelength values are.  Test 2: How much is the optical output power of the LEDs to determine the sanitization time. |
| Result 1: For the 3535 LEDs the peak wavelengths for the 265 nm was 275 nm and for the 275 nm it was 285 nm as shown in Figure 63.  Result 2: The optical power produced was by both the LEDs was between 0.6 mW to 0.8 mW. |
| Lens | Test 1: How much optical power of the LED is lost after passing through the lens. |
| Result 1: Power without lens – 2.9 mW  Power with lens - 0.5 mW |
| Beam Splitter  /Grating  /Dichroic Mirror | Test 1: Check if the beams can be combined. |
| Result 1: It was not possible to combine as the bandwidth difference was very small. |
| Photoresistor  /Phototransistor | Test 1: The photoresistor and phototransistor will be tested to determine whether it can, in conjunction with a fluorescent filter or alone, reliably detect UV wavelengths. |
| Result 1: Voltage reading with darkness, UV light on and UV light on with a filter used in front of the detector.  Photoresistor: 8 V, 7.02 V and 6 V without LEDs, with LEDs and with LEDs and filter respectively.  In the system the ADC readings are: 1 LEDs are off and 30 when LEDs are on.  Phototransistor: 0.112 V, 0.135 V and 0.185 V without LEDs, with LEDs and with LEDs and filter respectively. |
| Reed Switch | Test 1: Check if magnetic sensitivity of the switch works. |
| Result 1: The test worked perfectly and when the bottle cap is removed the reed switch activates and stops any task which is being performed. |

Table 48: Sanitizer Component Tests

UV-C LED Spectrums Measured by a Spectrometer

Chart

Description automatically generated

Chart, line chart

Description automatically generated

Figure 63: UV-C LED measured spectrum.

1. Showing the spectrum of 265 nm LED (b) Showing the spectrum of 275 nm LED

## Systems

Systems are multi-component constructs which perform the major functions of the project (such as sanitizing the bottle's contents, analyzing the bottle's contents for inorganic contaminants, or charging the battery). The tests for these systems are collected below (see Table 49). For more precise results from final FLASC tests, please see the testing section around 18:13 of our Final Powerpoint Presentation which can be found on our website at UCF.

Project System Tests

|  |  |
| --- | --- |
| System | Tests |
| Results |
| Analyzer | Test 1: The analyzer must be run using distilled water.  Test 2: The analyzer must be run using coffee.  Test 3: The analyzer must be run using 4 ppm fluoride. |
| Result 1: The minimum clear-water value was obtained for the ADC and set to 300.  Result 2: The extremum for a minimum water clarity was obtained and found to be less than 10.  Result 3: The system was run after full assembly (to verify that additional environmental light was not an issue) and verified to reject the fluoride mixture. |
| Sanitizer | Test 1: The efficiency of sanitizing the water by killing the pathogens. |
| Result 1: Using the Safewater Bacteria Test Strips, tested UCF’s Reflection Pond water before and after the sanitization test performed by the UVC LEDs (3 minutes of illumination of UV-C LEDs at 6V and 75 mA. The test proved that the LEDs sanitized the dirty water. |
| Microcontroller | Test 1: The current draw of the microcontroller during sleep mode and active mode will be measured with an in-series current sensor. This will further help improve battery life approximations and ensure that the microcontroller is properly entering and exiting low power mode. |
| Result 1: The test proved that the device properly performing low power modes. |
| Push Buttons | Test 1: Ensure that buttons properly send trigger event signal to microcontroller with simple test code to turn on LEDs. Test code was also be used to check button hold functionality |
| Result 1: Testing proved that both buttons work properly. A debouncing timer of 10 microseconds had to be added to avoid the decoupled button signals. |
| Battery | Test 1: The battery level was checked and recorded using the export app. Then the device was run through for 10 full cycles and the difference in battery level was recorded  Test 2: The battery was drained to a point where our microcontroller would not allow user operation and then the device was set on a Qi charging pad to measure the time to achieve a full charge wirelessly. |
| Result 1: Initial charge reading was 84%, charge after 10 cycles was 78%. Therefore, the device only lost ~6% of its charge after running for 10 full cycles. Result 2: The device began charging wirelessly at 17:20 and finished between 23:50 and 1:20. The total charge time was between 7.5-8 hours. |

|  |  |
| --- | --- |
| Voltage Regulators | Test 1: Connect regulator outputs to oscilloscope to measure nominal voltage levels and peak-to-peak values while unloaded |
| Result 1: Testing verified that regulators were properly operating and outputting voltages within a reasonable margin of error. |
| Bluetooth | Test 1: The data transfer time between the FLASC and app was tested.  Test 2: The device was operated at a distance of 20 ft and the Received Signal Strength Indicator was recorded along with the success of data transfer. |
| Result 1: The device is able to transfer the full payload to the app in well under 1 second.  Result 2: The device showed an RSSI of -79 dBm and was able to successfully transmit data |

Table 49: Project System Tests

## Programs

Programs are software applications which control project systems and perform computational analysis and processing of data output from the various systems.

While it was originally intended that programs be unit tested, the programs were instead tested in debugging mode as most issues were not purely software in nature.

### Servomotor Optical Demo Code (Obsolete)

For a optical system demonstration at the end of senior design one, it was requested that a simple program be created for manipulating the servomotor. The requested demo code defaulted to positioning the servomotor at 90 degrees, and then had implementation for manually adjusting the servomotor. Button 1 on the launchpad decremented the servomotor angle by approximately 1.7 degrees, and Button 2 increased the angle by the same amount. The servomotor was allowed to only move from 90 degrees to 100 degrees; therefore, the code prevented the servomotor from incrementing/decrementing past this range.

The first step was to configure P1.6 to be a timer module output. Next, the period of the timer was set as 20 ms. This was generated from SMCLK running at 1 MHz. The timer was utilized in up mode and programmed to count up to a value of 20,000-1. Thus resulting in a 20-millisecond period for the PWM signal, which matches the servomotor requirements laid out by the datasheet. Then a reset/set pattern channel was configured to create the 1.5 ms duty cycle. Of course, this code also required configuring the buttons and activating low power mode. Then the port 1 ISR determines which button was pressed and increment/decrements the duty cycle accordingly. Conditional statements were used to prevent the servomotor from moving outside of the requested range.

The PWM signal was observed on the oscilloscope and the waveforms exactly matched expected results. When the code booted up, the duty cycle is 1.5 ms and the period is 20 milliseconds. Each button press incremented the duty cycle by 10 microseconds (for the same period), and the servomotor did not travel outside of its angular range.

### Stepper Motor Code (Obsolete)

The stepper motor code did not actually interact with the stepper motor directly, but instead with the stepper motor driver, which added functionality (1/8 steps and sleep) to the system while reducing the number of pins on the PCB required for operation of the motor. The code on the PCB needed to control the Sleep, MS1, MS2, Step, and Dir pins on the driver board. The Sleep, MS1, and MS2 pins would have been always-high whenever the device was used, so these three pins were connected on the driver and plugged into the Sleep pin on the PCB. Prior to issuing any commands, the code set these pins high and waited ten microseconds to allow the board to wake up. The code used a high/low from the PCB's Dir pin to the driver's Dir to indicate whether the motor's polarity. The PWM pin on the PCB used a rising edge to signal a step through Step.

# Facilities and Equipment

This sections details some of the facilities and equipment which were used for the project, as specialized devices and tools were needed for manufacturing and testing prototypes as well as performing field research.

## Facilities

The facilities used to manufacture and test for the project are listed in Table 50: Facilities below.

Facilities

|  |  |  |
| --- | --- | --- |
| Location | Designation | Usage |
| CREOL A210 | PSE Senior Design Lab | Printing and prototyping the device housing as well as testing optical components. |
| CREOL 157 | Dr. Ivan Divliansky’s Lab | Prototyping and testing the sanitizer. |
| CREOL 240 | Optical Glass Cutting Lab | Cutting the (obsolete) prism into a smaller piece thanks to Dr. Mhibik. |
| ENG1 456 | ECE Senior Design Lab | Prototyping the embedded hardware and troubleshooting electrical issues. |
| I-CON Systems Inc. | Ryan’s Workplace | Using oscilloscopes, power supplies, and multimeters to test hardware functionality |

Table 50: Facilities

## Equipment

The equipment used to manufacture and test the project components and systems is detailed in Table 51 below. Common tools such as hacksaws, screwdrivers, and breadboards are omitted.

Equipment

|  |  |
| --- | --- |
| Device | Usage |
| Optical Wetsaw | Cutting the prism to meet the desired specs (obsolete) |
| Prusa 3D Printer | Creating the 3D printed cap casing to house the components of the project |
| Spectrometer | To measure the optical spectrum of the UV-C LEDs.  Evaluating the laser diode output. |
| Power meter | To measure the optical power intensity of the UV-C LEDs. |
| Multimeter | To test the currents and voltages flowing through the UV-C LEDs in different schematics. |
| Soldering machine | To solder wires to the UV-C LEDs so it can be implemented on the breadboard. |
| Safe home (Enviro Test Kits) | To check the presence of bacteria before and after sanitization. |

Table 51: Equipment

# Constraints and Standards

## Realistic Design Constraints

Constraints are limitations and requirements dictated by the customer, client, or environment at large. While the FLASC does not have a customer, the customer constraints were anticipated using the shared consumer experiences of the project leads. Environmental constraints both ecological and market-based are explored as well in this section.

### Economic and Time Constraints

When designing this product, economic constraints were some of the most crucial constraints to consider. This applied from both developmental and consumer standpoints. As far as design, this project was self-sponsored and therefore all team members preferred to try to minimize development and production costs as much as possible. This team has a very limited amount of capital to invest on this project and therefore sought to minimize wasteful purchases and/or decisions.

When sourcing any electrical or optical component, it was crucial to heavily factor in the cost for each decision. A more expensive component had to provide significant value in order to justify modifying the economic constraint. Another reason to keep the design as cost-effective as possible was because part of this project’s marketing goal was to be affordable for the average consumer. When conducting a cost-analysis of the current smart water bottle market, it was clear that there is room for a more affordable yet effective and capable smart water bottle. Minimizing project/development costs helped better meet this requirement to have a more marketable and attractive product.

Another way to minimize project costs was to utilize equipment and tools already available at UCF. This project required an extensive amount of test equipment for: signal analysis, water quality, sample preparation, and laser tuning. Fortunately, UCF provided labs, testing equipment, signal analysis equipment, and more to help mitigate these costs while ensuring proper device operation, although the chemistry department was entirely unhelpful and sample preparation was, thus, performed wholly on our own. In addition to this, utilizing development boards already obtained from previous classes (that utilize the same microcontroller) also slightly decreased development costs.

The project faced three separate time constraints: runtime, build time, and shipping time.

Product runtime had to be minimized as a bottle that takes a significant amount of time to sanitize and analyze would find its appeal drained before its contents. Similarly, a slow app and connection speed would result in product disinterest. In today's society, everyone is used to immediacy and being able to do, learn, or receive something nearly instantaneously. When designing this product, it wass important to keep this in mind and try to speed up device processes as much as possible. Speeding up certain processes, such as sanitization, would necessitate more powerful (and expensive) components, however.

Another important consideration was the short period of time allotted for the design of this project. Having a little over half a year to design a prototype was not a lot of time at all, not lessened by the significant excess administrative work that was required. Especially during the initial project phase, it was important not to waste any time but to instead ensure that every team member was keeping a solid pace. The team met twice weekly for the first semester to discuss project management and ensure that semi-weekly goals were being accomplished, although this tended towards weekly meetings in the second semester. This was also a valuable time for discussion and collaboration on overarching project ideas. This team worked on this project nearly every day to avoid falling behind schedule. Additionally, creating both tentative and concrete milestones had helped to motivate the team to accomplish many goals on a weekly basis. It is always easier to compartmentalize one larger milestone into smaller milestones (with mandatory deadlines) to ensure that development remains on schedule.

The market throughout both semesters was struggling amid supply-chain issues which were one of the biggest contributors to time constraints for this project. Semiconductors, and even batteries, were hard to come by. It was imperative that parts were ordered rapidly and arrived in expedient fashion so that they could be tested and implemented into the project design. Even board assemblers had significant delays, so it was crucial that the prototype boards were designed as quickly as possible to allow a significant amount of time for boards to arrive. Having this mindset of proactivity ultimately helped accelerate the development process- the sooner the boards arrived, the more time we had to program them and develop the software.

### Environmental, Social, and Political Constraints

The product addressed several environmental issues- the usage of disposable plastic bottles (which it seeks to reduce), the threat of waterborne diseases (which it seeks to eliminate), and the presence of toxic contaminants such as fluoride. That is not to say, however, that the device is inherently and invariably beneficial to the environment, society, or the political realm.

A potential environmental issue was present in the product's energy storage and usage. The product had to have a limited battery capacity to reduce the toll from lithium's damaging production methods while still remaining functional. Using components capable of low-power modes (such as the microcontroller) was also a must to increase the lifetime of the battery as well as reduce the amount of power needed for operation. Appropriate software modifications also had to be employed to reduce the power consumption of the device during operation. All of these considerations are summarized below in Table 52.

Environmental Constraints

|  |  |  |
| --- | --- | --- |
| Microcontroller | Sleep Mode Energy Consumption | < 30mA |
| Battery | Maximum Capacity | ≤ 2,500 mAh |
| Bottle | Material | Metallic, with the exception of a plastic or cork seal and upper lid |
| Overall Product | System durability | >50 ops​ |

Table 52: Environmental Constraints

The product also had social constraints, as certain design choices were liable to alienate large portions of society from the product. Economic constraints, which have been discussed previously, were certainly a major factor in social availability. Oher factors included color and part interchangeability, however. Device color is important as, although there is no universally agreed-upon 'favorite color', there are a few which stand out. To this end, the device was not colored yellow or orange, as these are generally considered the least-common favorite colors, but instead left the unoffending glossy black of uncolored PLA and the shiny neutrality of brushed stainless steel. Part interchangeability was key with regard to the bottle itself, as users were likely to drop and dent the bottle without damaging the device itself. To this end, the entire product (energy storage, wireless capability, charging, sanitization, analysis, and device control) was to be contained entirely inside the bottle's cap. The sole exception of this rule, during implementation, was the Reed Switch's magnet, which had to be secured to the bottle.

#### Restriction of Hazardous Substances Directive (RoHS)

RoHS was established to reduce the amount of Electrical/Electronic equipment waste that accumulates annually. This standard was originally established in the EU in an effort to better protect the environment and public health. Strict laws have been followed since 2003 that restrict the use of certain hazardous substances. The directive seeks to limit the use of the following ten substances: lead, cadmium, mercury, hexavalent chromium, polybrominated biphenyls (PBB) and polybrominated diphenyl ethers (PBDE), bis(2-ethylhexyl) phthalate (DEHP), butyl benzyl phthalate (BBP), dibutyl phthalate (DBP) and diisobutyl phthalate (DIBP). Currently, U.S. manufacturer must abide by this standard in order for their product to enter that particular market [69]. This standard in general, was a great one to follow for this project. It did not make sense for a project focused on reducing reusable bottles of water and water conservation to be producing electronics that are comprised of toxic materials. So, it was nice to know that PCBWay follows RoHS and therefore this project’s board is RoHS approved [70].

### Ethical, Health, and Safety Constraints

The project faced a potential ethical dilemma regarding assumed safety. Safeguards had to be in place to ensure that the sanitizer did, in fact, sanitize. This could be accomplished to a minimal degree simply by wiring a visible LED in series with the sanitizer LEDs. The visible LED can/will only shine if the sanitizer LEDs are operational. The project requires some constraints regarding health and safety, primarily for the UVC LEDs and laser, although the electrical components are also potentially hazardous. This implementation, however, would not have detected issues regarding decreased output light separate from electrical issues, so a feedback system was added into the device on Dr. Kar's recommendation to verify that the UVC LEDs were, in fact, functioning. This feedback system stops the sanitization and alerts the user if insufficient optical power is detected.

The UVC LEDs and laser had to be connected in series with a switch which ceased to function when the device was opened. This was satisfied using a Reed switch, although another implementation exploiting the metallic body constraint and using it as a terminal was also considered. The latter solution ran the risk of electrostatic damage and other undesirable phenomena, however. The UVC LEDs and lasers were thus contained in an opaque metal bottle which, barring intentional destruction, would not transmit any radiation.

The original intention of the device housing was to also provide a waterproof shell. While the housing is very nearly water-resistant and requires only a few additions (such as button covers and seals between layers), we did not have the time and means to fully waterproof the device while still maintaining accessibility to the components inside.

### Manufacturability and Sustainability Constraints

Several constraints governed the project for the sake of a more efficient and cost-effective production as well as sustainability in the market are examined in Table 53 below.

Manufacturing and Sustainability Constraints

|  |  |  |
| --- | --- | --- |
| Sanitizer | | |
| Cost​ | <$30​ | Maximum cost |
| Time​ | 2-3min ​ | Maximum time |
| Quality​ | 99.9% ​ | Effectiveness of sanitization |
| Analyzer | | |
| Cost​ | <$150​ | Maximum spectrometer budget (Obsolete) |
| Time​ | 1 minute​ | Maximum analysis runtime |
| Size​ | <36 in³​ | Maximum volume of prototype |
| Durability​ | >50 ops​ | Minimum number of expected operations |
| FLASC | | |
| Design Size​ | 4” diameter x 5” height | Maximum spacial requirements |

Table 53: Manufacturing and Sustainability Constraints

## Standards

The standards section covers standards the project intends to comply with. These standards provided enhanced interface capabilities with the broader market realm as well as performance benchmarks (see Table 54 below).

Standards

|  |  |  |
| --- | --- | --- |
| Bluetooth | | |
| RF Communication | The Bluetooth wireless standard is maintained by the Bluetooth Special Interest Group (SIG) [71] and the selection of wireless communication transmitter complied to the standard. | |
| Qi Wireless Charging | | |
| Induction Charging | The Qi Wireless charging standard is maintained by the Wireless Power Consortium [8], and the power receiver that was purchased for this project adhered to this standard. | |
| Battery Pack | | |
| Battery Composition | RoHS2: The Restriction of Hazardous Substances in Electrical and Electronic Equipment was put into force by the European Commission via the RoHS Directive on July 21, 2011 [72]. This law restricts the use of particular hazardous substances in electrical and electronic equipment. | |
| Sanitizer | | |
| UV-C light​ | The UV light when exposed is harmful for the human beings. ​ | Need to make it cost efficient so the price should not exceed this amount.​ |
| Safety​ | It will have a reed sensor on the bottle cap that will switch the laser off whenever the cap is opened. ​Use photoresistor for feedback of UVC’s | Need it to be set and working for this amount of time for proper efficiency. Need to know if the intensity of UVC’s is enough to kill the microorganisms. ​ |
| Wavelength​ | 265 nm – 295nm ​ | Need to be 99.9% efficient to kill all the microorganisms present.​ |
| Current​ | 75 mA​ | Should be fit into a limited space as we want it compact to fit into the bottle cap. |
| Voltage​ | 5.7 V​ | Need a way to protect the electronics from over-heating. ​ |
| Analyzer | | |
| Sensitivity​ | The spectrometer had to be capable of providing accurate test results for at least water and one contaminant at the concentration listed in the EPA regulatory standards [3].​ | |

Table 54: Standards

# Specifications

This section contains specifications on various utilized project components. Enough documentation is provided to allow for reproducibility of results. This part of the document also serves as a central location for the specific operating details of each and every device sub-system, namely Table 55 for the analyzer, Table 56 for the UVC sanitizer, and Table 57 for the power supply.

Analysis/Spectrometer Specifications

|  |  |  |
| --- | --- | --- |
| Component | Aspect | Specification |
| Servomotor  (Obsolete)​ | Size | 23 mm long by 12.5 mm wide by 22 mm tall |
| Angular Range​ | 180 degrees​ |
| Voltage | ~5 V |
| Current (unloaded) | 90 mA |
| Signal duration | .5-2.5 ms |
| Prism​  (Obsolete) | Principle Angle​ | Equilateral |
| Material​ | NBK-7​ |
| Size​ | ~5 mm³​ |
| Laser​ | Intensity​ | 100 mW |
| Voltage | 3.7 V (maximum) |
| Current | 250 mA (maximum) |
| Wavelength​ | 532 nm​ |
| Size | 12 mm in width and 35 mm height |
| Photodiode  (Obsolete)​ | Responsivity (at laser wavelength)​ | >= 0.3 A/W​ |
| Biasing Voltage | -30 V (maximum reverse bias) |
| Rise time | ~400 ns |
| Size | 5.4 mm diameter and 17.8 mm height |
| Mirrors​  (Obsolete) | Reflectance​ | >90% (at laser wavelength)​ |
| Size​ | ½” x ½”​ |
| Lens  (Obsolete) | Type | Fresnel Plano-Convex |
| Focal Length | 10 mm |
| Material | Acrylic |
| Size | 1/2" diameter and 1.5 mm height |

Table 55: Analysis/Spectrometer Specifications

Sanitizer Specifications

|  |  |  |
| --- | --- | --- |
| Component | Aspect | Specification |
| LED | Quantity | 2 |
| Type | SMD LED |
| Peak Wavelengths | 274 nm and 283 nm |
| Voltage | 6 V |
| Current | 75 mA total for both placed in parallel |
| Intensity | 6 -15 mW |
| Material | AlGaN-based Flip Chip |
| Mass | ~83 mg each |
| Size | 3.6 mm-3.6 mm, 1.71mm thick |
| PCB size | 10 mm diameter, 1 mm thick |
| PCB Material | Copper |
| Photoresistor | Wavelength | 540 nm |
| Operating Temperature | -300 C to 700 C |
| Response Time | 30 ms |

Table 56: Sanitizer Specifications

Power Supply Specifications

|  |  |  |
| --- | --- | --- |
| Component | Aspect | Specification |
| Battery | Capacity | 2,500 mAh |
| Type | Lithium Polymer |
| Mass | 43 g |
| Nominal Voltage | 3.7 V |
| Size | 1.8" x 2.4" x 0.26" |
| Charger | Max Charging Current | 500 mA |
| Pass Transistor ON-Resistance | 0.35 Ω |
| Power Receiver | Type | Inductive |
| Output Voltage | 4.8V – 5.2V |
| Current | 500 mA |
| Size | 1.2” x 1.8” x 0.01” |

Table 57: Power Supply Specifications

# Structural Diagram

The project device was very constrained by space even with the admittedly large design. As can be seen in an early structural diagram below (see Figure 64), it did appear that the components chosen, which are to-scale in the diagram, would have been physically capable of fitting inside the proposed design. It is worth noting that a production-model device would have used significantly more compact, purpose-built components, leading to a much smaller (and cheaper) device- a good example of such a product would be a CD reader/writer, which has a complex optical path, mechanical components controlling the optical path with extreme precision, and feedback all at a low cost and small size.

The final implementation can be explored directly using the 3D model files available on the website or at the project's [GitHub](https://github.com/SmartWaterBottleProject/Device-Housing).

Structural Diagram (Obsolete)

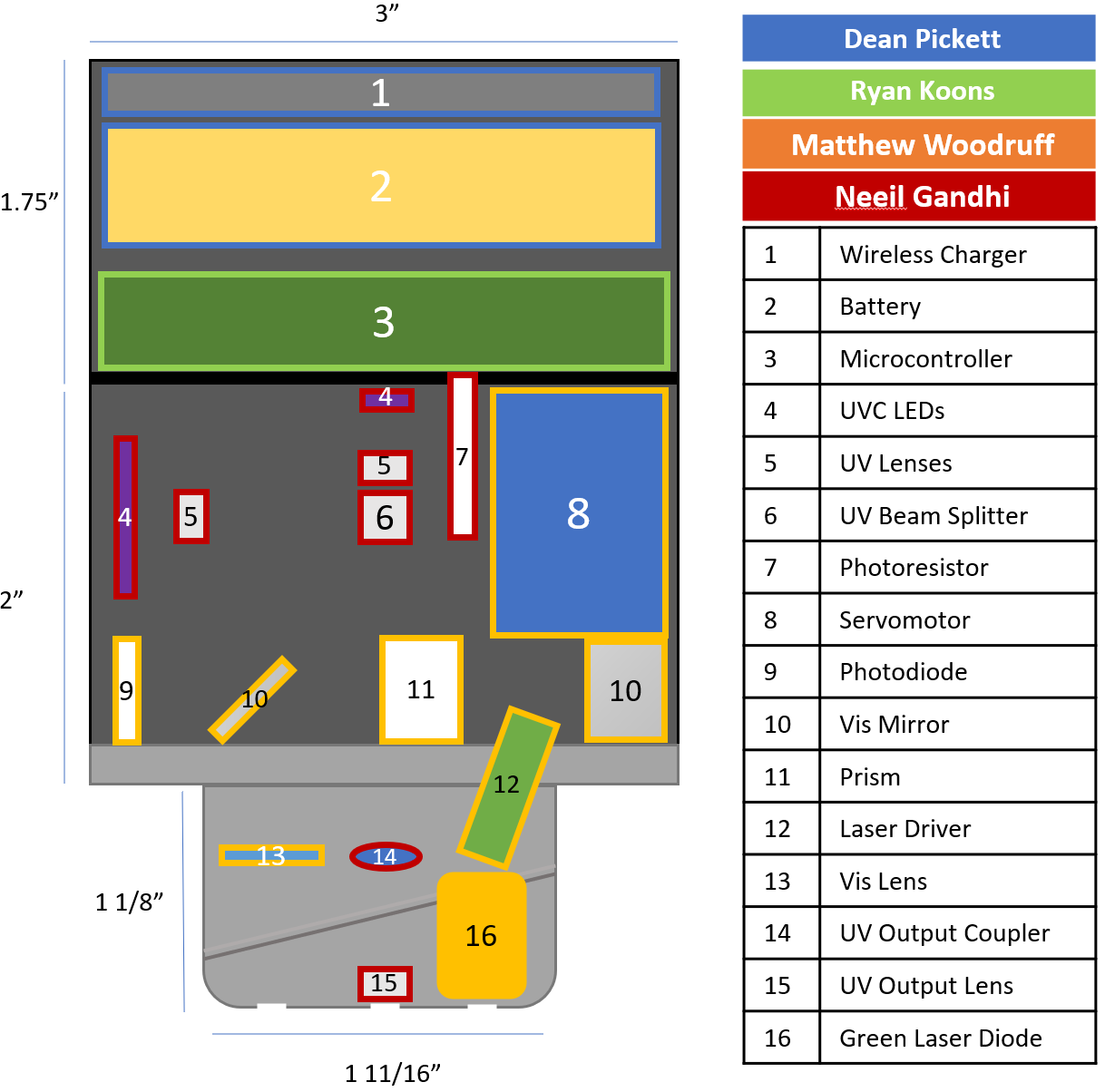


Figure 64: Structural Diagram

# Biographies

|  |  |
| --- | --- |
| A person smiling for the camera  Description automatically generated with medium confidence | Ryan Koons is graduating and receiving his Bachelor’s of Science degree in Electrical Engineering in December of 2021. From here, he will be pursuing a career as an “Electrical Engineer I” at CAE USA Inc. in Tampa, Florida. He loves drumming, listening to music, and spending time with his friends and family. |
| A person with a beard and glasses  Description automatically generated with medium confidence | Matthew Woodruff is on course to graduate with a Bachelor's of Science degree in Photonic Science and Engineering in December 2021 and intends to find employment locally. He enjoys cooking, horticulture, programming, and tinkering with projects. |
| A person with a mustache  Description automatically generated with low confidence | Neeil Gandhi is an international student from India who will be graduating in December 2021 with a Bachelor’s of Science degree in Photonic Science and Engineering. After graduating he will be going to work at General Medical Engineering Company in Erlangen, Germany. In his free time, he enjoys dancing, reading, playing sports and spending quality time with friends and family. |
| A person with a beard  Description automatically generated with medium confidence | Dean Pickett is scheduled to graduate in December 2021 with a Bachelor of Science degree in Electrical Engineering. His degree track is Power and Renewable Energy, and he hopes to find local employment in the power industry. He enjoys traveling with friends and family. |

# Summary and Conclusions

As one might have expected, there were many adjustments made in the technical approaches used for solving each problem, and a series of hurdles morphed the project significantly from its original vision- however, the fundamental goals of this project were achieved. The FLASC realizes the abilities it was created for- the device is capable of sanitizing its contents within a reasonable time frame and can analyze its contents for harmful contaminants (although it was only proven with fluoride). The FLASC can display this information on a smartphone app that is connected via Bluetooth Low Energy. The extensive research and initial development phases for this device were fruitful over the course of Senior Design 1, and while the concepts gathered during said course were frequently changed when converted from brainwaves into tangible objects, the technological investigation was, overall, a success.

The investigation into power transmission that was carried out has shown that the preliminary plans of creating a simple inductive charging setup between two coils that were wound by ourselves and operating at the 60 Hz provided right out of a wall socket would have been needlessly dangerous, and this would potentially have overcharged the Lithium-Polymer battery in the unit. The particular chemistry of battery was an outstanding option on the market due to the higher charge storage per unit volume and the lower price with respect to the charge capacity. Therefore, extensive considerations were made to accommodate this type of energy storage. The most pronounced choice in this department was the use of premade power transmitters and receivers that are compliant with the Qi wireless power transmission standard. In terms of regulating the voltage supply rails for the device components, it was decided that using a switch-mode Boost topology would give the most efficiency for the devices requiring more power and being unaffected by the voltage ripple that is inherent in these designs. Meanwhile, despite many efforts made to design a 3V switch-mode converter for the digital components, photodiode, and photoresistor, it was decided in the end that it is of utmost importance to optimize this converter for voltage regulation. This meant that the Buck-Boost design that was initially proposed was given up in favor of a Low Dropout Linear Voltage Regulator that would not have an output ripple from the device switching.

The research investigation involved in water sanitization system had to go through multiple steps. Starting with which method would best fit our project and whether it was safer using only one method or multiple methods. Additional decisions had to be made on what UV source to use with what wavelengths and the optical design for such, then sanitization testing and, lastly, a feedback mechanism to ensure the proper operation of our device. There was a lot of growth and learning in this area as we researched more into it, which had to be suited for the project design constraints such as cost, size, and energy consumption. Taking all the things into account the decisions we made have been verified to be at least acceptable, at least by the rapid coliform testing used.

The water quality analyzer was, unfortunately, abandoned following equipment issues made worse by shipping delays. It remains unclear whether the design could be made functional using the components chosen, although the photodiode for one seems to have had too low of a responsivity to have effectively worked in the system. That is not to say that no milestones were passed on the design- before the design was dropped, the optical layer had adjustable mirrors custom-designed which functioned as intended, could extract a tight beam of light from the bottle and align it through a prism and mirrors, and feasibly had the space needed for the remaining components. It would have been preferable to prototype intermediary designs earlier in SD II or even late in SD I, but the lack of acquired expertise in CAD made such a goal difficult to attain, although it would be within grasp now.

The water clarity analyzer, which was hastily designed and replaced the previous water quality analyzer, was a surprisingly effective system and accomplished one of the previous design's goals rather effectively with a low-cost and low-complexity approach. Were this product to be redesigned and pushed to market, it seems worthwhile to investigate this mock-infrared spectrometry further as a low-cost implementation.

Looking into the topic of safety for the device use, there was initially some contention about how the microcontroller would be able to detect whether or not the cap is secured onto the bottle in order to prevent accidental laser operation while the cap is off. It seemed like a good option to use a pressure activated switch that would be pressed while the cap was secured, but it was deemed too easy for a user to accidentally engage this sort of safety mechanism with their fingers while handing the cap. Consequently, we turned towards switches that could be actuated by the presence of a magnetic field because these sorts of devices are becoming increasingly more common in smart devices. The question about whether to use a Hall effect or Reed switch was resolved rather quickly, because of the component simplicity and lower power requirements. The biggest concession that had to be made with this decision was that the bottle would need to have a magnet attached near the lip, where a user would drink from.

There was also a great deal of effort put into research regarding the Bluetooth module that would be used for this project. Due to the power limitations of having a battery that can fit in a water bottle cap, it was highly desirable to select a Bluetooth Low Energy device that would only transmit data whenever it was necessary to communicate with the smartphone app. The Cypress module that was found had out-of-the-box connectivity capabilities with the CySmart app, which allowed us to focus on how the microcontroller would send data to the module via UART.

During SD I we discovered that we had access to 3D printers in the CREOL senior design lab, which later proved imperative to the completion of the project, and this was unanimously deemed to be perfect for our needs in terms of creating a custom housing. While our original goal of attaching a plastic housing to the threaded cap was unsuccessful, entirely replacing the metal region as well proved an effective design, although the loss of a reflective base is something which would need to be addressed in a commercial product, likely by simply adding a circular, reflective plate to the base with holes for the optical inputs and outputs. The PLA cap worked extremely well with the wireless charger, keeping only a 4 mm separation between the receiver and transmitter while also not producing eddy currents within the cap.

For the embedded hardware design, the goal was to design a schematic that was flexible, simple, and functional. Flexibility in the schematic/board design eliminated the need for future board revisions which was critical considering the time it took to have them made. The design process began by creating sub-systems to help organize and compartmentalize each part of the printed circuit board. From here, project components were extensively researched by all team members in order to come up with the best design. This process also required a large amount of microcontroller operation research. After technological investigation, the pin configuration had to be selected for the desired microcontroller. This pin configuration needed to be able to work with both the final microcontroller and the development board. The pin configuration was created so that the traces on the final product would be easy to follow. From here an initial schematic was created. This required numerous conversations/discussions about how each system would interface with the microcontroller. Each sub-system required considering power consumption requirements, how it would interface with the board, and how it would attach to the microcontroller. Many devices/symbols had to be created in EAGLE in order to represent project components in the schematic. A fair amount of MOSFET switching logic also had to be figured out in order to determine how to properly enable/disable different project requirements. In addition to this, less integral product components had to be selected and their datasheet had to be inspected. These components also had to be available in the market when we ordered them.

While this is the end of the road for the FLASC itself, the concepts, and executions of said concepts will remain in the acquired knowledge and skills of the team members as well as, in a less metaphorical sense, in the UCF database.

# References

1. Connor, Bradley A. (2019) “Travelers' Diarrhea - Chapter 2 - 2020 Yellow Book,” *Centers for Disease Control and Prevention*. [Online]. Available: <https://wwwnc.cdc.gov/travel/yellowbook/2020/preparing-international-travelers/travelers-diarrhea>. [Accessed: 08-Jun-2021].
2. National Research Council (US) Safe Drinking Water Committee. “Drinking Water and Health” Volume 2*. Washington (DC): National Academies Press (US); 1980*. Available from: https://www.ncbi.nlm.nih.gov/books/NBK234592/ doi: 10.17226/1904. [Accessed 10-Jun-2021].
3. Environmental Protection Agency, “National Primary Drinking Water Regulations,” *Environmental Protection Agency*. [Online]. Available: <https://www.epa.gov/ground-water-and-drinking-water/national-primary-drinking-water-regulations>. [Accessed: 10-Jun-2021].
4. Larason, T. (2020), NIST Transportable Tunable UV Laser Irradiance Facility for Water Pathogen Inactivation, Review of Scientific Instruments, [online], <https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=930308>. [Accessed June 10, 2021].
5. Swagatam. (2019) “3v, 4.5v, 6v, 9v, 12v, 24v, Automatic Battery Charger Circuit with Indicator,” *Homemade Circuit Projects*. [Online]. Available: <https://www.homemade-circuits.com/3v-45v-6v-9v-12v-24v-automatic-battery/>. [Accessed: 10-Jun-2021].
6. “Inductive Versus Resonant Wireless Charging: A Truce May Be a Designer's Best Choice,” DigiKey, 02-Aug-2016. [Online]. Available: [https://www.DigiKey.com/en/articles/inductive-versus-resonant-wireless-charging](https://www.digikey.com/en/articles/inductive-versus-resonant-wireless-charging). [Accessed: 22-Jun-2021].
7. P. K. D. A. 28, “What is Eddy Current Loss? - definition and expression,” Circuit Globe, 28-Nov-2019. [Online]. Available: <https://circuitglobe.com/what-is-eddy-current-loss.html>. [Accessed: 22-Jun-2021].
8. E. Notes, “Qi Wireless Charging Standard,” Electronics Notes. [Online]. Available: <https://www.electronics-notes.com/articles/equipment-items-gadgets/wireless-battery-charging/qi-wireless-charging-standard.php>. [Accessed: 23-Jun-2021].
9. L. Ada, “Li-Ion &amp; LiPoly Batteries,” Adafruit Learning System. [Online]. Available: <https://learn.adafruit.com/li-ion-and-lipoly-batteries/voltages>. [Accessed: 24-Jun-2021].
10. “A Designer's Guide to Lithium (Li-ion) Battery Charging,” DigiKey, 01-Sep-2016. [Online]. Available: [https://www.DigiKey.com/en/articles/a-designer-guide-fast-lithium-ion-battery-charging](https://www.digikey.com/en/articles/a-designer-guide-fast-lithium-ion-battery-charging). [Accessed: 24-Jun-2021].
11. “LDO ＜What is an LDO?＞,” ROHM. [Online]. Available: https://www.rohm.com/electronics-basics/dc-dc-converters/what-is-ldo. [Accessed: 01-Jul-2021].
12. I. Batarseh and A. Harb, Power Electronics Circuit Analysis and Design. Cham: Springer International Publishing, 2018.
13. R. Bearson, “What is the difference between Bluetooth versions?” *Ear Rockers*, 19-Mar-2021. [Online]. Available: <https://earrockers.com/difference-between-bluetooth-versions/>. [Accessed: 25-Jun-2021].
14. “Learn About Water,” *Water Quality Association*. [Online]. Available: <https://www.wqa.org/Learn-About-Water/Common-Contaminants>. [Accessed: 10-Jun-2021].
15. CDC, "Drinking Water Treatment Methods for Backcountry and Travel Use," 20 February 2009. [Online]. Available: <https://www.cdc.gov/healthywater/pdf/drinking/Backcountry_Water_Treatment.pdf>.
16. "Technology/Support" *SETi | Technology | UV LED*. SETi Sensor Electronic Technology, Inc. [Online]. Available: <http://www.s-et.com/en/technology/uvled/> . [Accessed: 17-Jun-2021].
17. Tissue Optics and Photonics: Biological Tissue Structures - Scientific Figure on ResearchGate. Available from: <https://www.researchgate.net/figure/Electromagnetic-spectrum-and-types-of-interaction-with-matter-UV-ultraviolet-EUV_fig1_288887087>. [Accessed 24 Jun, 2021].
18. "Everything You Should Know About UVC LED Light (2020 Update)." *ShineLong*. 04-Aug-2020. [Online]. Available: <https://www.shinelongled.com/uvc-led-light/>. [Accessed: 20-Jun-2021].
19. Bergeron, Bryan. "UV Sanitizer: How to Build One and Measure Its Efficacy." *Nuts and Volts Magazine. [Online].* Available: <https://www.nutsvolts.com/magazine/article/uv-sanitizer-how-to-build-one-and-measure-its-efficacy>. [Accessed: 22-Jun-2021].
20. Center for Devices and Radiological Health. "UV Lights and Lamps: Ultraviolet-C Radiation, Disinfection, and Corona." *U.S. Food and Drug Administration*, 2021 [Online]. Available: <https://www.fda.gov/medical-devices/coronavirus-covid-19-and-medical-devices/uv-lights-and-lamps-ultraviolet-c-radiation-disinfection-and-coronavirus>. [Accessed: 23-Jun-2021].
21. “Technology/Support,” *SETi | Technology | UV LED*. [Online]. Available: <http://www.s-et.com/en/technology/uvled/>. [Accessed: 01-Jul-2021].
22. O. Lawal, J. Cosman, and J. Pagan, “UV-C LED Devices and Systems: Current and Future State,” *IUVA News*, AquiSense Technologies LLC, 2018. Available: <https://uvledsource.org/wp-content/uploads/54-UVC-LED-Devices-and-Systems.pdf>. [Accessed: 25-Jun-2021].
23. Craig Moe, “UV-C Light Emitting Diode,” *RADTECH REPORT*, 2014. [Online]. Available: <https://www.radtech.org/magazinearchives/Publications/RadTechReport/mar-2014/UV-C%20Light%20Emitting%20Diodes.pdf>. [Accessed: 28-Jun-2021].
24. “UVC 275nm LED modules, services and strengths of Universal Science,” *Universal Science*, 11-Jun-2021. [Online]. Available: <https://www.universal-science.it/en/moduli-led-uvc-i-servizi-e-i-punti-di-forza-di-universal-science/>. [Accessed: 15-Jul-2021].
25. Djhamer, “UV LED Exposure Box,” *Instructables circuits*, 08-Nov-2017. [Online]. Available: <https://www.instructables.com/UV-LED-Exposure-Box/>. [Accessed: 15-Jul-2021].
26. T. C. Larason, “National Institute of Standards and Technology transportable tunable ultraviolet laser irradiance facility for water pathogen inactivation,” *AIP Publishing*, 01-Jul-2020. [Online]. Available: <https://aip.scitation.org/doi/10.1063/5.0016500>. [Accessed: 15-Jul-2021].
27. J. R. Bolton, I. Mayor-Smith, and K. G. Linden, “Rethinking the Concepts of Fluence (UV Dose) and Fluence Rate: The Importance of Photon‐based Units – A Systemic Review,” *Wiley Online Library*, 24-Sep-2015. [Online]. Available: [https://onlinelibrary.wiley.com/doi/10.1111/php.12512#](https://onlinelibrary.wiley.com/doi/10.1111/php.12512). Accessed: 15-Jul-2021].
28. S. E. Beck, H. B. Wright, R. A. Rodriguez, K. G. Linden, T. M. Hargy, and T. C. Larason, “Wavelength Dependent UV Inactivation and DNA Damage of Adenovirus as Measured by Cell Culture Infectivity and Long Range Quantitative PCR,” *ACS Publications*, 22-Nov-2013. [Online]. Available: <https://pubs.acs.org/doi/pdf/10.1021/es403850b>. [Accessed: 15-Jul-2021].
29. J. Ferraro and K. Nakamoto, “Introductory Raman Spectroscopy,” 2nd edition. Elsevier, 2003.
30. T. Collette and T. Williams, “The Role of Raman Spectroscopy in the Analytical Chemistry of Potable Water.” *Royal Society of Chemistry*, 2002 [Online]. Available: <https://pubs.rsc.org/en/content/articlehtml/2002/em/b107274a>. [Accessed: 25-Jun-2021].
31. J. Bridgeman, A. Baker, D. Brown, and J.B. Boxall, “Portable LED Fluorescence Instrumentation for the Rapid Assessment of Potable Water Quality.” *Science of the Total Environment*, 2015 [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S004896971500501X>. [Accessed: 25-Jun-2021].
32. P. Liang, Y. Qin, B. Hu, T. Peng, and Z. Jiang, “Nanometer-size Titanium Dioxide Microcolumn On-line Preconcentration of Trace Minerals and their Determination by Inductively Coupled Plasma Atomic Emission Spectrometry in Water.” *Analytica Chemica Acta*, 2001 [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0003267001010108>. [Accessed: 25-Jun-2021].
33. R. Karka, P. Kumar, B. Bansod, and C. Krishna, “Analysis of Heavy Metal Ions in Potable Water Using Soft Computing Technique.” *Procedia Computer Science*, 2016 [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0003267001010108>. [Accessed: 25-Jun-2021].
34. THORLABS, “Right-Angle Prisms,” *THORLABS*, 2021 [Online]. Available: <https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=142>. [Accessed: 25-Jun-2021].
35. THORLABS, “UV Ruled Reflective Diffraction Gratings,” *THORLABS*, 2021 [Online]. Available: <https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=26>. [Accessed: 25-Jun-2021].
36. R. Paschotta, “Image Sensors,” *RP Photonics Encyclopedia*, 2021 [Online]. Available: <https://www.rp-photonics.com/image_sensors.html>. [Accessed: 25-Jun-2021].
37. R. Christ, R. WernliSr., "Video," *The ROV Manual*, 2014 [Online]. Available: <https://www.sciencedirect.com/topics/engineering/composite-video-signal>. [Accessed: 04-Jul-2021].
38. RP Photonics, "Photodetectors," *RP Photonics*, 2021 [Online]. Available: <https://www.rp-photonics.com/photodetectors.html>. [Accessed: 03-Jul-2021].
39. LAFVIN, "LAFVIN 5 Sets 28BYJ-48 ULN2003 5V Stepper Motor + ULN2003 Driver Board for Arduino," LAFVIN, 2021 [Online]. Available: <https://www.amazon.com/LAFVIN-28BYJ-48-ULN2003-Stepper-Arduino/dp/B076KDFSGT/ref=sr_1_11_sspa?dchild=1&keywords=micro+stepper+motor&qid=1625523294&sr=8-11-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyVkNSSFRXMjFSUFQmZW5jcnlwdGVkSWQ9QTAzMzgxOTUzQUlSMTZIWllETjY2JmVuY3J5cHRlZEFkSWQ9QTAyMjc5NjAyTFBDTFhTNUtKTzBHJndpZGdldE5hbWU9c3BfbXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ==>. [Accessed: 05-Jul-2021].
40. NW, "5pcs 6V dDia 10mm Micro 2 Phase 4 Wire Stepper Motor 18 Degress Mini Stepping Motor," NW, 2021 [Online]. Available: <https://www.amazon.com/Micro-Phase-Stepper-Degress-Stepping/dp/B07CSLQK11/ref=sr_1_9?dchild=1&keywords=micro+stepper+motor&qid=1625523294&sr=8-9>. [Accessed: 05-Jul-2021].
41. AUKUYEE, " AUKUYEE Plastic Gear Set, 75Pcs Single Double Reduction Gear Worm Gear for DIY Car Robot QY12," AUKUYEE, 2021 [Online]. Available: <https://www.amazon.com/Quimat-Plastic-Single-Double-Reduction/dp/B06XCG24HZ/ref=sr_1_2?dchild=1&keywords=gear+assortment&qid=1625523703&sr=8-2>. [Accessed: 05-Jul-2021].
42. Deegoo-FPV, "4Pcs SG90 9g Micro Servos for RC Robot Helicopter Airplane Controls Car Boat," Deegoo-FPV, 2021 [Online]. Available: <https://www.amazon.com/Micro-Servos-Helicopter-Airplane-Controls/dp/B07MLR1498/ref=sr_1_1_sspa?dchild=1&keywords=servomotor&qid=1625529856&sr=8-1-spons&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUFJMTNSTU5PMDVVUkImZW5jcnlwdGVkSWQ9QTA1NzY0MzMyUFMzVjE1N0VIVVY1JmVuY3J5cHRlZEFkSWQ9QTAzMDM1MDMxR01XRjFNMkU3Q0VEJndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ&th=1>. [Accessed: 05-Jul-2021].
43. OSI Optoelectronics, “Multi Element Photodiode Array,” *OSE Optoelectronics*, 2020 [Online]. Available: <https://www.osioptoelectronics.com/standard-products/silicon-photodiodes/photodiode-arrays/multi-element-photodiode-array.aspx>. [Accessed: 25-Jun-2021].
44. S. Writer, “How to Decide Between a Reed Switch or a Hall Switch,” Thomasnet® - Product Sourcing and Supplier Discovery Platform - Find North American Manufacturers, Suppliers, and Industrial Companies, 12-Oct-2018. [Online]. Available: https://www.thomasnet.com/insights/how-to-decide-between-a-reed-switch-or-a-hall-switch/. [Accessed: 30-Jun-2021].
45. J.M.K.C. Donev et al. (2018). Energy Education - Permeability of free space [Online]. Available: https://energyeducation.ca/encyclopedia/Permeability\_of\_free\_space. [Accessed: July 21, 2021].
46. P. Bishop, “A tradeoff between microcontroller, DSP, FPGA and ASIC technologies,” *EETimes*, 25-Feb-2009. [Online]. Available: <https://www.eetimes.com/a-tradeoff-between-microcontroller-dsp-fpga-and-asic-technologies/>. [Accessed: 23-Jun-2021].
47. TronicsZone Editorial Staff, “FPGA Vs Microcontroller: When to Use What?,” *TronicsZone*, 23-Feb-2021. [Online]. Available: <https://www.tronicszone.com/blog/fpga-vs-microcontroller/>. [Accessed: 23-Jun-2021].
48. “RF Wireless World,” *Advantages of DSP | disadvantages of DSP*. [Online]. Available: <https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-DSP.html>. [Accessed: 24-Jun-2021].
49. M. Short, “Getting Started with the Raspberry Pi Zero Wireless,” Sparkfun. [Online]. Available: https://learn.sparkfun.com/tutorials/getting-started-with-the-raspberry-pi-zero-wireless/all. [Accessed: 04-Jul-2021].
50. A. Industries, “Raspberry Pi Zero W,” adafruit industries blog RSS. [Online]. Available: https://www.adafruit.com/product/3400?gclid=CjwKCAjwuIWHBhBDEiwACXQYsQuOmsvoJeAfSCp9c6R994S6nTL\_el9GU4fg\_l0NYBmI9rv9BqZZEBoCxwAQAvD\_BwE. [Accessed: 04-Jul-2021].
51. J. Yiu and I. Johnson. (Mar. 2013). The Many Ways of Programming an ARM® Cortex®-M Microcontroller. [Online]. Available: https://community.arm.com/cfs-file/\_\_key/telligent-evolution-components-attachments/01-1989-00-00-00-00-52-02/The-many-ways-of-programming-an-ARM-Cortex\_2D00\_M-microcontroller.pdf
52. A. Nguyen, “Bluetooth 1.0 vs 2.0 vs 3.0 vs 4.0 vs 5.0 - How They Compare | Symmetry Blog,” *SymmetryElectronics.com*, 18-Apr-2018. [Online]. Available: https://www.semiconductorstore.com/blog/2018/Bluetooth-1-0-vs-2-0-vs-3-0-vs-4-0-vs-5-0-How-They-Differ-Symmetry-Blog/3147/. [Accessed: 25-Jun-2021].
53. Bates, Matthew. (2013) “Build Your Own Induction Charger,” *Nuts & Volts Magazine*. [Online]. Available: <https://www.nutsvolts.com/magazine/article/august2013_Bates>. [Accessed: 05-Jun-2021].
54. “EZ-SERIAL™: EZ-BLE MODULE FIRMWARE PLATFORM,” Cypress.com, 21-Jun-2021. [Online]. Available: https://www.cypress.com/documentation/software-and-drivers/ez-serial-ez-ble-module-firmware-platform. [Accessed: 06-Jul-2021].
55. “Microchip Bluetooth Data - Apps on Google Play,” Google, 02-Jun-2021. [Online]. Available: https://play.google.com/store/apps/details?id=com.microchip.bluetooth.data. [Accessed: 07-Jul-2021].
56. J. Katsandres, “Bluetooth Low Energy -It Starts with Advertising,” Bluetooth® Technology Website, 15-Feb-2017. [Online]. Available: https://www.bluetooth.com/blog/bluetooth-low-energy-it-starts-with-advertising/. [Accessed: 16-Jul-2021].
57. E. Pena and M. G. Legaspi, “UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter,” Analog Dialogue, Dec-2020. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html. [Accessed: 07-Jul-2021].
58. “macOS Big Sur is compatible with these computers,” Apple Support, 17-Nov-2020. [Online]. Available: https://support.apple.com/en-us/HT211238. [Accessed: 25-Jun-2021].
59. “Meet Android Studio,” Android Developers. [Online]. Available: https://developer.android.com/studio/intro. [Accessed: 25-Jun-2021].
60. “Top Programming Languages for Android App Development,” GeeksforGeeks, 04-Mar-2021. [Online]. Available: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/. [Accessed: 25-Jun-2021].
61. “Lesson: Object-Oriented Programming Concepts,” Oracle: The Java™ Tutorials; Learning the Java Language. [Online]. Available: https://docs.oracle.com/javase/tutorial/java/concepts/index.html. [Accessed: 19-Jul-2021].
62. Ellisys Bluetooth Video 5: Generic Attribute Profile (GATT). Ellisys, 05-Jun-2018. *YouTube*. [Online] Available: https://www.youtube.com/watch?v=eHqtiCMe4NA. [Accessed: 17-July-2021].
63. C. Y. Ong, “The Ultimate Guide to Android Bluetooth Low Energy,” Punch Through, 15-May-2020. [Online]. Available: https://punchthrough.com/android-ble-guide/. [Accessed: 17-Jul-2021].
64. “Documentation: ScanFilter,” Android Developers, 24-Feb-2021. [Online]. Available: https://developer.android.com/reference/android/bluetooth/le/ScanFilter. [Accessed: 19-Jul-2021].
65. “Asynchronous programming techniques,” Kotlin, 03-Jun-2021. [Online]. Available: https://kotlinlang.org/docs/async-programming.html. [Accessed: 20-Jul-2021].
66. L. Klint, "Azure DevOps vs GitHub: Comparing Microsoft's DevOps Tools," A Cloud Guru, 20-Jan-2021. [Online]. Available: <https://acloudguru.com/blog/engineering/azure-devops-vs-github-comparing-microsofts-devops-twins>. [Accessed: 21-Jul-2021].
67. P. A. Ladanyi and S. M. Morrison, “Ultraviolet Bactericidal Irradiation of Ice,” *Applied Microbiology*, vol. 16, no. 3, pp. 463–467, 1968.
68. UFO Battery, "Does Cold Weather Affect Lithium-ion Battery?," UFO BATTERY, 18-Dec-2019. [Online]. Available: <https://www.ufo-battery.com/does-cold-weather-affect-lithium-ion-battery>. [Accessed: 19-Jul-2021].
69. “RoHS Directive,” European Commission. [Online]. Available: https://ec.europa.eu/environment/topics/waste-and-recycling/rohs-directive\_en. [Accessed: 09-Jul-2021].
70. “RoHs Lead Free,” Custom PCB Prototype Manufacturer. [Online]. Available: https://www.pcbway.com/pcb\_prototype/RoHs\_Lead\_Free.html. [Accessed: 09-Jul-2021].
71. “Board of Directors,” Bluetooth® Technology Website. [Online]. Available: https://www.bluetooth.com/about-us/board-of-directors/. [Accessed: 09-Jul-2021].
72. European Commission. 2021. *RoHS Directive*. [online] Available at: <https://ec.europa.eu/environment/topics/waste-and-recycling/rohs-directive\_en> [Accessed 22 September 2021].

# Acknowledgements

The group would like to thank Dr. Divliansky for letting us use his lab and equipment such as spectrometer, power supply and soldering machine to test and design our system.

We would also like to thank Dr. Mhibik for assisting us with cutting a flint-glass prism (which was part of the spectrometer implementation).

Ryan would like to thank Blaine Andersen from I-CON Systems for all of his advice and assistance regarding embedded hardware and software design. Ryan’s work supervisor, Blaine, was very helpful with checking conceptuality behind device schematics earlier on in the development process. He assisted with potential design issues such as buttons not working and needing to develop a method for limiting the power consumption of the device when the cap was removed. Blaine also helped recommend replacement components throughout the multitude of part shortages that required redesigns. Ryan personally feels that he was better equipped for this senior design project because of all of the experience and wisdom that he obtained from Blaine.

# Appendix A: Datasheets

1. [MS-2431-3 Press-Fit Sensor Datasheet](https://www.pic-gmbh.com/fileadmin/user_upload/datasheets_en/ms24313_ds_e_1.2.pdf)
2. [SMCO5 Magnet Datasheet](http://4a30d8fd18dae1bf393d-df49f4cedb726ad03ad145d2e3d346bd.r41.cf5.rackcdn.com/datasheets/40/4003004017e.pdf)
3. [TVP5150AM1 Ultralow-Power NTSC/PAL/SECAM Video Decoder datasheet (Rev. E) (ti.com)](https://www.ti.com/lit/ds/symlink/tvp5150am1.pdf)
4. [ATmega48A, ATmega48PA, ATmega88A, ATmega88PA, ATmega168A, ATmega1688PA, ATmega328, ATmega328P datasheet (microchip.com)](https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf)
5. [MSP430G2x53, MSP430G2x13 Mixed Signal Microcontroller datasheet (Rev. J) (ti.com)](https://www.ti.com/lit/ds/symlink/msp430g2553.pdf?ts=1625512622275&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP430G2553)
6. [MSP430FR698x(1), MSP430FR598x(1) Mixed-Signal Microcontrollers datasheet (Rev. D) (ti.com)](https://www.ti.com/lit/ds/symlink/msp430fr6989.pdf?ts=1625503912298&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP430FR6989%253Futm_source%253Dgoogle%2526utm_medium%253Dcpc%2526utm_campaign%253Depd-null-null-GPN_EN-cpc-pf-google-wwe%2526utm_content%253DMSP430FR6989%2526ds_k%253D%257B_dssearchterm%257D%2526DCM%253Dyes%2526gclid%253DCj0KCQjw24qHBhCnARIsAPbdtlJsmZnMl5NQxvH3cJ_YUvIR8oTA55Y_pAI5J2c7lE_V3_wn93zcGhsaAmHcEALw_wcB%2526gclsrc%253Daw.ds)
7. [MSP430F552x, MSP430F551x Mixed-Signal Microcontrollers datasheet (Rev. P) (ti.com)](https://www.ti.com/lit/ds/symlink/msp430f5529.pdf?ts=1626228549036&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP430F5529)
8. [CYBLE-013025-00 BLE Module Datasheet](https://www.mouser.com/datasheet/2/100/CYBLE-0130XX-00_EZ-BLE_WICED_MODULE-1375365.pdf)
9. [RN4870/71 Data Sheet](https://ww1.microchip.com/downloads/en/DeviceDoc/RN4870-71-Data-Sheet-DS50002489E.pdf)
10. [EYSPBNZUA Data Sheet](https://www.yuden.co.jp/jp/product/category/module/img/TY_BLE_EYSPBNZUA_DataReport_V1_0_20201019E.pdf)
11. [EZ-Serial WICED BLE Firmware Platform User Guide](https://www.cypress.com/file/408286/download)
12. [LIPO785060 2500mAh 3.7V Datasheet](https://cdn-shop.adafruit.com/product-files/328/LP785060+2500mAh+3.7V+20190510.pdf)
13. [MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family User's Guide (Rev. P) (ti.com)](https://www.ti.com/lit/ug/slau367p/slau367p.pdf?ts=1626992688586&ref_url=https%253A%252F%252Fwww.google.com%252F)
14. [APFA2507LSURKSYKZGKC(Ver.3A) (kingbrightusa.com)](https://www.kingbrightusa.com/images/catalog/SPEC/APFA2507LSURKSYKZGKC.pdf)
15. [AV02-0587EN DS ASMT-CB00 05May2010.pdf (broadcom.com)](https://docs.broadcom.com/doc/AV02-0587EN)
16. [pts840.pdf (ckswitches.com)](https://www.ckswitches.com/media/1477/pts840.pdf)
17. [MIC 5365/6 High-Performance Single 150mA LDO Datasheet](https://www.mouser.com/datasheet/2/268/mic5365-1082359.pdf)
18. [EZ-BLETM MODULE ARDUINO EVALUATION BOARD](https://www.cypress.com/file/389231/download)

# Appendix B: Copyrights and Permission

Permission for Figure 15: Electromagnetic Spectrum Figure taken from [17]:

Graphical user interface, text, application, email

Description automatically generated

Permission for Figure 16 and Figure 18 parts (a) and (b):

Graphical user interface, text, application, email

Description automatically generated

Permission for Figure 17 and Figure 18 part (c):

Graphical user interface, text, application

Description automatically generated

Permission for Figure 19:

Graphical user interface, text, application, email

Description automatically generated

# Appendix C: Project Spending

Project Spending

|  |  |
| --- | --- |
| Component | Cost |
| Device Housing | |
| Bottle | $13.99 |
| Custom Housing | $0 |
| **Total:** | **$13.99** |
| Bluetooth Module | |
| Cypress Bluetooth CYBLE-013025-EVAL Board (+tax & ship) | $54.51 |
| **Total:** | **$54.51** |
| Power | |
| 4 Batteries, 4 JST connectors, 4 LiPo chargers, 2 Qi receivers (+tax & ship) | $140.74 |
| **Total:** | **$140.74** |
| Sanitization | |
| LED 275 nm (including sales tax and shipping) (3 of them) | $16.53 |
| LED 265 nm (including sales tax and shipping) (3 of them) | $14.03 |
| Bacteria Test Kit (10 Strips) | $24.00 |
| **Total:** | **$54.56** |
| Spectrometer | |
| Prism (Obsolete) | $19.98 |
| Photodiode (Obsolete) | $14.58 |
| Laser diode | $16.15 |
| Phototransistor | $7.49 |
| Servomotor (Obsolete) | $11.99 |
| Stepper Motor (Obsolete) | $5.99 |
| Stepper Motor Driver (Obsolete) | $5.95 |
| Mirrors (Obsolete) | $8.99 |
| Lens (Obsolete) | $18.83 |
| Shipping + Tax | $20.62 |
| **Total:** | **$130.57** |
| Embedded Hardware | |
| Microcontrollers (MSP430FR5989IRGC x6) from TI | $38.62 |
| PCB (x5) and Assembly (x4) | $297.00 |
| Reed Switches (x4) | $13.64 |
| **Total:** | **$349.26** |
| ***Grand Total:*** | **$743.63** |

Table 58: Project Spending