Towards the Automation of Scalable Carbon Nanotube Sorting via Aqueous Two-Phase Extraction

by

Marcel Robitaille

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

© Marcel Robitaille 2022

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Marcel Robitaille is the sole author for the text of this thesis, which was written under the supervision of Dr. Na Young Kim. He is also the author of all figures unless otherwise specified in the figure's caption. Research contributed by others is enumerated in this section as well as being clearly specified in the text.

Dr. Na Young Kim and HeeBong Yang were instrumental in the planning and direction of this project. Marcel Robitaille was the primary researcher for this project. He was solely responsible, with some exceptions listed below, for all programming, 3D design, 3D printing, machining, mechanical assembly, circuit design, wiring, soldering, modification of off-the-shelf products, and documentation. He also contributed to the testing of the automated system. Dr. Na Young Kim, HeeBong Yang, Luke Neal, Matt Piatt, Nicki Caswell, and Garen Simpson contributed to lab work and testing of the automated system. These testers also programmed "recipes", using the software libraries developed by the author to write short scripts executed by the automated system. Ardhendu Shekhar Bhattacharya contributed mechanical designs, which he tested himself by programming small *Arduino* applications (Section 4.4.2). Zidong Xie, Muskan Lalit, and Lu Wang contributed code for the computer vision portion of the project. However, the final computer vision code was solely written by the author (Section 4.3.2). Dr. Na Young Kim and Luke Neal soldered backup printed circuit boards. Some machining was outsourced to the University of Waterloo Engineering Machine Shop (Section 4.1.2). Roberto Romero supported by assisting with the use of the IQC Tech Lab's 3D printers, by training the author in the use of the IQC Tech Lab's 5-axis CNC mill, and by constructing a table for the robotic system (Section 5.1).

Abstract

Carbon nanotubes are an exciting quantum material with numerous interesting and unique properties. Notable electric properties are their variable band-gap and metallic and semiconducting variants. Carbon nanotubes are a prime candidate for beyond-silicon devices and have already been used in experimental computer processors. Carbon nanotubes have valuable optical, chemical, and mechanical properties as well.

Many of these novel properties such as the band-gap and absorption spectrum depend on the carbon nanotube's chirality, which is related to its diameter. Unfortunately, no method for carbon nanotube synthesis has yet been able to produce pure, single-chirality nanotubes. Applications requiring a specific chirality must thus sort the synthesised nanotubes after the fact. The most scalable and affordable sorting method is aqueous two-phase extraction (ATPE). However, this procedure requires significant time and skill as it involves many stages of micro-pipetting.

The subject of this thesis is the automation of ATPE. Since this is a water-based procedure, the core around which the automated system is built is an off-the-shelf liquid handling robot. The majority of this text describes the custom extensions to this robot that make it capable of ATPE. The *Photo Stage* provides an environment to capture consistent images of the two-phase system, and the accompanying computer vision software analyses the image and locates the phase boundary. This information is used to programmatically extract the top phase. Transportation of vials is accomplished by the addition of the *Gripper*. The caps are removed from vials and tubes using the *Vial Opener*. The *Powder Handling System* has the goal of automating the creation of chemical stocks. The *Inventory System* is designed to track the contents and history of ATPE process vials. Finally, a robotic arm will integrate external instruments including a centrifuge, sonicator, and spectrophotometer. However, this is not complete and is planned for the second generation of the project.

Without the integration of the centrifuge, which is required at every step of ATPE to partition the solution into two phases, it is impossible to carry out ATPE from start to finish. Thus, the system is being used in a mode referred to as hybrid ATPE. Under these conditions, when the centrifuge is needed, the system is paused so that a human operator can transfer the vials to the centrifuge. When the centrifuge has finished, the vials are transferred back to the liquid handling robot and the system is resumed. This hybrid system takes advantage of the *Gripper*, *Vial Opener*, and *Photo Stage*. The *Inventory System* is incomplete in that the camera has not been mounted, however the associated code can already be used to track the contents of vials if the identifier is entered manually rather

than automatically being read from a barcode or other marker. The *Powder Handling System*, which is for the pre-processing step of stock-making and not for the main ATPE process, is not complete. A precision balance has been purchased and a conveyor has been developed, but there have been multiple failed attempts to fabricate a powder dispenser. Thus, the *Powder Handling System* is not used and stocks are still created manually. Finally, the external robotic system is not complete and is not used for hybrid ATPE.

Under hybrid ATPE, it has been possible to automatically extract the top phase from two-phase systems. The reliability of the computer vision boundary recognition system has been increased to $> 95\,\%$. However, the results of automated separation leave room for improvement. The robotic system leaves more discarded waste around the boundary than when ATPE is performed manually, and there is a heightened risk of contaminating the boundary. The reason for this is that the pipettes of the liquid handling robot are always vertical, parallel to the vial. This prevents the liquid that wicks up the vial walls by capillary action from being properly extracted. This will be addressed by future additions to the robotic system.

Acknowledgements

I would like to thank the members of my thesis committee, my advisor Prof. Na Young Kim, Prof. Nasser Lashgarian Azad, and Prof. Michael Pope, for allocating their time to be on my committee, to read my thesis, and to give feedback and insight. I would also like to thank Prof. Kim in her capacity as my advisor.

I would like to thank HeeBong Yang for his patience and selflessness. HeeBong always made time in his busy schedule to answer my questions about the background and chemistry of this project. Furthermore, his contributions in the lab were invaluable. It was an honour to work alongside HeeBong in the lab. I would also like to thank other lab members and URA students who contributed research, development, or lab work to this project: Nicki Caswell, Muskan Lalit, Luke Neal, Matt Piatt, Garen Simpson, Lu Wang and Zidong Xie. Additionally, I would like to thank other lab members and URA students, past and present, that I have had the pleasure of interacting with, including Aaron Gross, Anya Houk, Angéline Lafleur, Mai Sakuragi, and Henry Xi.

I would like to thank Roberto Romero from the *University of Waterloo IQC Tech Lab* for his assistance with 3D printing, machining, and construction. Roberto is a kind and patient teacher, and his enthusiasm about these technologies is infectious. I would like to thank the *Waterloo RoboHub*, especially Brandon and Robert, for their assistance in selecting, transporting, and testing a robotic arm for this project. Their insights and expertise were invaluable. I would like to thank the *University of Waterloo Engineering Student Machine Shop*, especially Andrew and Jeff, for teaching me to use a mill and lathe and assisting me in machining many parts for this project.

Finally, I would like to thank my parents, my sister, my extended family, my significant other, and close friends for showing me constant support and love when I needed it the most. I would like to thank my dear friend Aaron Gross for his advice on thesis-writing, his guidance navigating degree completion, and his emotional support. This degree would not have been possible without this support network. I am eternally grateful.

Table of Contents

List of Figures xii

List of Tables xvii

List of Abbreviations xviii

1 Introduction 1

2 Carbon Nanotube Background 3

2.1 Carbon Nanotube History and Theory 3

2.2 Methods of Carbon Nanotube Synthesis 7

2.2.1 Electronic Arc Discharge 9

2.2.2 Laser Ablation 10

2.2.3 Chemical Vapour Deposition 11

2.2.4 Gas Phase Chemical Vapour Deposition 12

2.3 Methods of Carbon Nanotube Sorting 13

2.3.1 Historical Sorting Processes 15

2.3.2 Aqueous Two-phase Extraction 16

2.3.2.1 Pre-processing Steps 21

2.3.2.2 Main ATPE Process 22

2.3.2.3 Post-processing Steps 25

3 Project Outline **27**
3.1 Problem Definition . . . 27
3.2 Plan . . . 27
3.2.1 Liquid Handling Robot Selection . . . 28
3.2.2 *OT-2* Liquid Handling Robot . . . 28

4 Custom Hardware and Software Components **31**
4.1 *Gripper* . . . 36
4.1.1 *Gripper V1* . . . 36
4.1.2 *Gripper V2* . . . 40
4.1.2.1 Feedback for the Angle of the Fingers . . . 44
4.1.2.2 Mechanical Assembly . . . 46
4.1.2.3 Printed Circuit Board and Wiring . . . 46
4.1.2.4 Software . . . 50
4.2 *Vial Opener* . . . 52
4.2.1 Mechanical Design and Assembly . . . 52
4.2.2 Printed Circuit Board and Wiring . . . 60
4.2.2.1 Mistakes . . . 62
4.2.3 Software . . . 63
4.2.4 Coordination with the *Gripper* . . . 64
4.3 Separation of the Two Phases . . . 66
4.3.1 *Photo Stage* . . . 66
4.3.1.1 Enclosure . . . 68
4.3.1.2 Camera . . . 70
4.3.1.3 Backlight . . . 70
4.3.1.4 Diffuser . . . 71
4.3.1.5 Opening Mechanism . . . 71
4.3.1.6 Printed Circuit Board . . . 72

4.3.1.7 Software . . . 74
4.3.1.8 Camera Drift Tracking . . . 75
4.3.1.9 Conversion Between Volume and Height . . . 77
4.3.1.10 Conversion Between Millimetres and Pixels . . . 78
4.3.2 Separation Computer Vision . . . 78
4.3.2.1 Statement of Contributions . . . 79
4.3.2.2 Attempt # 1: Vector-LabPics Pre-Trained Models . . . 79
4.3.2.3 Attempt # 2: MobileNet-SSD and Adaptive Thresholding 80
4.3.2.4 Attempt # 3: Semantic Segmentation Model . . . 83
4.3.2.5 Attempt # 4: Canny Edge Detection . . . 84
4.3.2.6 Attempt # 5: Improved Boundary Detection Algorithm . 86
4.3.2.7 Experiments and Results . . . 92
4.4 *Powder Handling System* . . . 96
4.4.1 Plan . . . 96
4.4.2 Powder Dispenser . . . 97
4.4.2.1 Attempt # 1: Sliding Powder Dispenser . . . 97
4.4.2.2 Attempt # 2: Archimedes Screw . . . 106
4.4.2.3 Attempt # 3: Vibrator and Mesh . . . 113
4.4.3 *Conveyor* . . . 115
4.4.4 *Weigh Module* . . . 120
4.5 *Inventory System* . . . 122
4.5.1 Identifier Format and Generation . . . 123
4.5.2 *Barcode Scanner* . . . 124
4.5.2.1 Preliminary Testing . . . 124
4.5.2.2 Integration . . . 126
4.5.3 *Barcode Printer* . . . 127
4.5.4 Tracking Code . . . 127

4.6 Software . . . 129
4.6.1 Firmware and Communications . . . 130
4.6.1.1 Communications Library . . . 130
4.6.1.2 Asynchronous Acceleration-Capable Stepper Motor Library 138
4.6.1.3 AS5600 Magnetic Induction Rotary Encoder Library . . . 149
4.6.2 Libraries for use in Opentrons protocols . . . 150
4.6.2.1 Liquid Library . . . 150
4.6.2.2 *Gripper*-Aware Pipette . . . 152
4.6.3 Manual control interface . . . 154
4.6.3.1 Interactive Command-Line Interface . . . 154
4.6.3.2 Implementation and architecture . . . 157

5 Integration of External Instruments 159
5.1 Robot Selection . . . 159
5.2 *OT-2* Integration . . . 161
5.3 Centrifuge Integration . . . 162
5.4 Sonicator Integration . . . 164
5.5 Spectrophotometer Integration . . . 165

6 Experiments, Testing, and Results 166
6.1 Hybrid ATPE . . . 166
6.1.1 Statement of Contributions . . . 166
6.1.2 Hybrid ATPE Results . . . 167
6.2 Status of Each Component . . . 170

7 Conclusion 175
7.1 Practical Lessons . . . 176
7.2 Future Prospects . . . 176

Bibliography 178

Appendices 197

A Schematic of the *Gripper V2* PCB in *Autodesk Eagle* 198

B Board of the *Gripper V2* PCB in *Autodesk Eagle* 204

C Schematic of the AS5600-based magnetic induction encoder PCB in *Autodesk Eagle* 206

D Board of the AS5600-based magnetic induction encoder PCB in *Autodesk Eagle* 208

E Schematic of the *Vial Opener* PCB in *Autodesk Eagle* 210

F Board of the *Vial Opener* PCB in *Autodesk Eagle* 216

G Schematic of the *Photo Stage* PCB in *Autodesk Eagle* 218

H Board of the *Photo Stage* PCB in *Autodesk Eagle* 225

I Schematic of the *Powder Handling Sliding Dispenser* PCB in *Autodesk Eagle* 227

J Board of the *Powder Handling Sliding Dispenser* PCB in *Autodesk Eagle* 229

List of Figures

2.1 Visual representations of exemplary armchair, zigzag, and chiral SWCNTs. 6

2.2 Visualisation of exemplary chirality vectors, unit vectors, and chiralities on a hexagonal grid representing the bonds of a graphene sheet. 7

2.3 A photograph of many purified single-chirality carbon nanotube samples showing the relationship between colour and chirality. 8

2.4 Schematic of an electric arc discharge chamber. 10

2.5 Schematic of a laser ablation CNT synthesis process. 11

2.6 Illustration of the chemical vapour deposition process. 12

2.7 Illustration of the HiPco® SWCNT synthesis procedure. 13

2.8 Illustration of the CoMoCAT® SWCNT synthesis procedure. 14

2.9 Schematic of sequential ATPE separations, forming a binary tree. 17

2.10 An exemplary polymer-polymer phase diagram. 18

2.11 Chemical structure of the surfactants used for ATPE. 19

2.12 Partition coefficient diagrams showing how to select the concentration of SDS. 20

2.13 Simplified illustration of the effect of the SDS concentration on the partitioning of the two phases. 20

2.14 Exemplary curve showing the approximate SDS concentration required to partition the SWCNT to the top phase. 21

2.15 Schematic of a typical generalized ATPE recipe for SWCNTs synthesised by CoMoCAT® and HiPco®. 24

2.16 Absorption spectra of sorted SWCNTs with annotations showing the primary chirality in each. 26

3.1 Image of the *Opentrons OT-2*. 29
3.2 Four exemplary deck layouts showing multiple configurations of first-party elements including tip racks, well plates, and modules. 30

4.1 Representation of the star architecture communication between the modules of the automation system. 33
4.2 A 3D image of the *OT-2* deck, showing the slot numbering system and the ridges used for positioning. 34
4.3 Planned layout of the modules on the deck of the *OT-2*. 35
4.4 Visual aid of the intended position of the *Gripper V1* inside the *OT-2*. . . 37
4.5 Side view of the *Gripper V1* showing that its fingers can extend below the tip of the pipette. 38
4.6 Side view of the *Gripper V1* showing the *Gripper* fingers, lead screw, stepper motor, sled, and timing belt. 39
4.7 Mounting location of the *Gripper V2* inside the *OT-2* behind the pipettes attached to the head of the machine. 41
4.8 Side view of the *Gripper V2* mounting position. 42
4.9 Internal components of the *Gripper V2* with annotations indicating the most important components. 43
4.10 Illustration of the difference between the two shafts of the *Gripper V2*. . . 45
4.11 Image of the magnetic induction encoders and I^2C multiplexer attached to the rear of the *Gripper V2*. 46
4.12 A photo of the *Gripper V2* installed behind the head of the *OT-2* after the mechanical assembly was completed. 47
4.13 A photo of the populated PCB for the *Gripper V2*. 48
4.14 Image of the 3D-printed cable chains used to route the cables required by the *Gripper V2*. 49
4.15 Image of two magnetic induction encoder position sensor PCBs. 50
4.16 CAD screenshot of the *Vial Opener*. 54
4.17 Annotated index of the pieces of each sub-assembly of the *Vial Opener* module. 55
4.18 Cross sectional view of the mechanism to grasp and release the vial. 56

4.19 Cross sectional view showing the cup of the *Vial Opener* in each supported configuration. . . . 57
4.20 A photo of the fully-assembled *Vial Opener*. . . . 58
4.21 CAD screenshot of the *Vial Opener* with cap storage addition. . . . 59
4.22 Image of the PCB wired into and installed inside the *Vial Opener*. . . . 61
4.23 Oscilloscope traces illustrating the need for a flyback diode in the circuit disconnecting the solenoid. . . . 63
4.24 Image of the internals of the *Photo Stage* showing the major components. . 67
4.25 Image of a sample taken inside the *Photo Stage* using the Raspberry Pi Camera Module. . . . 68
4.26 The position of the *Photo Stage* inside the *OT-2*. . . . 69
4.27 Image of the *Photo Stage* camera mount with 3 degrees of freedom. . . . 70
4.28 The opening mechanism of the *Photo Stage* featuring a servo motor. . . . 72
4.29 A photo of the populated PCB for the *Photo Stage*. . . . 73
4.30 A photo of the *Photo Stage* PCB wired to the servo motor and backlight. . 74
4.31 An example of the ArUco-based camera drift tracking. . . . 76
4.32 Result of the calibration of the conversion between volume and height. . . 77
4.33 Result of the calibration of the conversion between pixels of the *Photo Stage* camera and millimetres of the *OT-2* pipette. . . . 79
4.34 Illustration of the progression of 3 models to recognize first all vessels, then each vessel individually, then the contents of each vessel. . . . 80
4.35 Visualisation of the difference between global and adaptive thresholding on an example image of an ArUco code. . . . 81
4.36 Visualisation of the difference between global and adaptive thresholding on an exemplary image of a vial. . . . 82
4.37 The result of Muskan Lalit's method on an exemplary image of a vial. . . . 82
4.38 The result of the semantic segmentation boundary detection method. . . . 83
4.39 The result of the Canny edge detection boundary recognition method. . . . 85
4.40 Visualisation of each possible outcome of the result of the Canny boundary detection algorithm. . . . 86

4.41 Visualisation of the method to find the strongest boundary in the Canny image. . . . 88
4.42 The result of Canny edge detection on an exemplary sample with two dark and opaque phases. . . . 89
4.43 The result of the compression colour averaging method on an exemplary sample with two dark and opaque phases. . . . 90
4.44 The result of thresholding edge detection on an exemplary sample with two dark and opaque phases. . . . 91
4.45 Flowchart showing the order of steps in the improved boundary detection algorithm. . . . 93
4.46 Validation image for the only remaining incorrectly detected two-phase system with two very clear phases. . . . 94
4.47 Closeup of the meniscus between the two phases. . . . 95
4.49 CAD screenshots of the refined sliding powder dispenser. . . . 99
4.50 CAD screenshots of the sliding dispenser's actuating mechanism. . . . 100
4.51 A photo of the populated PCB for the *Powder Handling Sliding Dispenser*. 102
4.52 Illustration of the results of the experiment testing the refined sliding powder dispenser design. . . . 103
4.53 Images of the gauge used to quickly find the correct sliding dispenser clearance. 104
4.48 Images of the prototype sliding powder dispenser. . . . 105
4.54 Cross-sectional view of the Archimedes screw powder dispenser CAD. . . . 107
4.55 Array of four Archimedes screw powder dispensers in their mounts. . . . 108
4.56 Photos of the Archimedes screw powder dispenser where the screw and housing are printed using the MSLA printed in photopolymer resin. . . . 109
4.57 Dramatized illustration of the warped Archimedes screw housing. . . . 110
4.58 The fabrication of the Archimedes screw using the *Pocket NC* CNC machine. 111
4.59 A collection of Archimedes screws and housings manufactured using various fabrication technologies. . . . 112
4.60 Images showing the build up of SC powder while testing the Archimedes screw dispenser. . . . 113

4.61 CAD screenshot of the designed tube cap with a grid of small 1 mm square holes. . . . 114
4.62 Image of the 1 mm, 0.75 mm, and 0.5 mm caps printed in resin. . . . 115
4.63 A screenshot of the CAD model of the *Conveyor* used in the *Powder Handling System*. . . . 116
4.64 Cross-sectional view of the link and tube, featuring a taper at the bottom. 117
4.65 A screenshot of the CAD model of the cog with multiple links. . . . 118
4.66 A CAD screenshot showing how the bottom plate interfaces with the deck of the *OT-2*. . . . 119
4.67 Image of the fully-assembled conveyor. . . . 120
4.68 Image of the balance in the *OT-2*. . . . 121
4.69 Results of preliminary tests of 2D QR codes. . . . 125
4.70 Diagram of messages between the *Central Command* module and the microcontroller for various scenarios. . . . 136
4.71 Change in angular velocity ω from one step $(n-1)$ to the next (n) with constant acceleration $\dot{\omega}$. . . . 139
4.72 Flowchart of the logic to take and schedule steps. . . . 141
4.73 Sequences of steps to update the stepping period for the next step. . . . 145
4.74 A simulation of the constant acceleration equations. . . . 146
4.75 Screenshot of a modified version of the Opentrons app to support updating the volumes from the GUI. . . . 153
4.76 Screenshot of the *Manual Control Interface*. . . . 155

5.1 Image of the *Franka Emika Panda*. . . . 160
5.2 Sketch of the planned placement of instruments on the robotics table. . . . 161
5.3 Two images of 4 correctly detected ArUco codes. . . . 164
5.4 Image of the grid of holes to repeatably hold tubes in the sonicator. . . . 165

6.1 Experimental data showing the approximate SDS concentration required to partition the SWCNT with a given chirality into the top phase. . . . 168
6.2 Spectrophotometer spectra of purified SWCNT samples produced by hybrid ATPE. . . . 169

List of Tables

2.1 List of stocks showing chemicals and target concentrations. 22

4.1 Command names, codes, and functions specific to the *Gripper V2*. 51
4.2 Variants of the *Gripper V2* status enumeration. 52
4.3 Command names, codes, and functions specific to the *Vial Opener*. 64
4.4 Variants of the *Vial Opener* status enumeration. 64
4.5 Command names, codes, and functions specific to the *Photo Stage*. 75
4.6 Tuning parameters of the improved phase boundary detection algorithm. . 92
4.7 Example identifier for a process vial and meaning of each block. 123
4.8 Example identifier for a stock and meaning of each block. 124
4.9 Format of the SQLite table tracking all information for ATPE main process. 128
4.10 Format of a command message. 131
4.11 Format of a response message. 132
4.12 Variants of the "response code" enumeration. 132
4.13 Command names, codes, and functions of the base commands. 133
4.14 Public methods of the stepper motor library class. 148
4.15 Variants of the stepper library status enumeration type. 149
4.16 Public interface of the magnetic induction rotary encoder library. 149
4.17 The functionality provided by each tile of the *Manual Control Interface*. . 156

6.1 Time required by the *OT-2* for each step of a bottom extraction recipe. . . 170
6.2 Status of each module of the automation system. 171

List of Abbreviations

ATPE aqueous two-phase extraction iv, v, vii, x, xii, xvi, xvii, 1, 2, 14, 16–19, 21–25, 27–29, 31, 32, 65, 66, 77, 83, 85, 88, 92, 94, 96, 97, 110, 120, 122–124, 127–129, 150, 154, 159, 166, 167

C carbon 4, 13

CAD computer-aided design xiii–xvi, 37, 43, 45, 46, 54, 59, 99, 100, 105, 107, 114, 116, 118, 119

CNC computer numerical control xv, 110–112

CNT carbon nanotube xii, 1, 3–23, 122, 128

CO carbon monoxide 13

CO_2 carbon dioxide 13

CoMoCAT® cobalt–molybdenum catalyst xii, 12–14, 23, 24

CRI colour rendering index 71

CSV comma-seperated values 151

CVD chemical vapour deposition xii, 11, 12

DNA deoxyribonucleic acid 14–16, 18

DOC sodium deoxycholate 15, 18–23, 25, 96, 97, 128, 129, 167

DWCNT double-walled carbon nanotube 6

DX dextran 16–18, 21, 22, 96, 128, 129

EMS engineering machine shop 46

FDM fused deposition modeling 28, 98, 110, 112

GUI graphical user interface 32, 78, 129, 151, 152, 162

HiPco® high pressure carbon monoxide xii, 12, 13, 23, 24

I²C inter-integrated circuit xiii, 45, 46, 48, 49, 149, 150

IQC Institute for Quantum Computing vi, 110

JSON JavaScript object notation 152

LED light-emitting diode 70, 71

MSLA masked stereolithography xv, 98, 100, 106, 108, 109, 112, 114

MWCNT multi-walled carbon nanotube 4, 9, 21

NaClO sodium hypochlorite 18

PCB printed circuit board xi, xiii–xv, 29, 46, 48–50, 60–62, 72–74, 101, 102

PEG polyethylene glycol 16–18, 21, 22, 96, 98, 102, 112, 114, 127–129

PLA polylactic acis 69, 112

PWM pulse-width modulation 72, 73, 101

QR code quick response code xvi, 75, 77, 92, 124–128, 163

REPL read-eval-print loop 131, 154

ROS robot operating system 162

SC sodium cholate xv, 15, 18–23, 96, 97, 102, 103, 112–114, 128, 129

SDS sodium dodecyl sulfate xii, xvi, 18–23, 96, 97, 128, 129, 167, 168

SSD single-shot multi-box detection ix, 80, 82

SWCNT single-walled carbon nanotube xii, xvi, 4, 6, 7, 9, 12–16, 19–21, 23–27, 123, 167, 168

TCP transmission control protocol 152, 158, 162

UART universal asynchronous receiver-transmitter 48, 50, 60, 63, 73, 74, 101, 130, 131, 134, 148

URA undergraduate research assistantships vi, 78, 97

USB universal serial bus 121, 122, 127

UV-Vis ultraviolet-visible 25, 166

Chapter 1

Introduction

The term "robotics" sparks various images for different people. The inventor remembers one of their creations coming to life. The futurist imagines getting work done while their car drives them to the office. The accountant thinks of cost-savings, while the operator fears that their livelihood may become obsolete. These are all valid perspectives, though it is doubtful that many researchers will resent the vial-cleaning robot that frees them to focus on pushing the boundaries of science. This is one of the many reasons that robotics is a good fit for the laboratory.

Many lab processes are time-consuming, error-prone, and uninspiring. A robot can complete these tasks more efficiently, with nearly perfect repeatability, and nearly zero complaints. Furthermore, the higher throughput that can be achieved with an automated process opens the door of entirely new possibilities. Large-scale experiments that could not be dreamed of with samples meticulously prepared by hand are now within reach.

This thesis details the automation of carbon nanotube sorting using the aqueous two-phase extraction (ATPE) method. Chapter 2 presents the properties, applications, history, synthesis methods, and sorting methods of carbon nanotubes (CNTs). This background information justifies the appeal of CNTs and explains why they must be sorted for certain applications. Chapter 3 formalises the problem definition and presents the plan to automate this sorting process. Chapter 4 details the numerous custom hardware and software components developed over the course of this project that serve to automate ATPE. These include a *Gripper*, a *Vial Opener*, a *Photo Stage*, a *Powder Handling System*, and an *Inventory System*. The planned next generation of this project, which involves the inclusion of a robotic arm that will enable the incorporation of a centrifuge, a sonicator, and a spectrophotometer, is discussed in Chapter 5. The testing and experimental results

of this automation system are discussed in Chapter 6, where it is revealed that mostly-autonomous ATPE is possible, although the yield is not perfect. The only step of the main ATPE process still requiring manual intervention is moving the samples to and from the centrifuge. Finally, a conclusion is given in Chapter 7.

Chapter 2

Carbon Nanotube Background

This chapter gives background information on the exciting properties of carbon nanotubes, justifying the existence of this project. A history and theory of CNTs is given in Section 2.1. The most common synthesis methods are presented in Section 2.2, while Section 2.3 justifies the need for carbon nanotube sorting and describes the sorting method used in this project.

2.1 Carbon Nanotube History and Theory

Carbon nanotubes (CNTs) are an exciting quantum material with countless applications. They can present both metallic and semiconductor-like electrical properties, making them of great interest to nano-scale circuit designers and positioning them as a prime candidate for next-generation beyond-silicon processors. Indeed, modern processors are already at the limit of transistor density in silicon. Carbon nanotube-based processors address this by enabling much higher theoretical densities. Early CNT processors have already been demonstrated [1]–[3]. Further electrical properties of CNTs include their high electroconductivity [4], superb radiation efficiency [5], and scattering-free spin-conserving transport of electrons [6]. Additionally, unlike most materials, semiconducting CNTs have varying bandgaps, which is highly desirable in certain applications. This material has interesting optical properties as well, such as optical absorption, photoluminescence in the near-infrared region, and strong resonance Raman scattering, rendering CNTs effective for imaging [4], [7]–[12]. Additionally, CNTs provide desirable chemical properties, notably chemical inertness and high hydrogen absorption capacity [13], [14]. Finally, these nanoparticles also have very appealing mechanical properties. CNTs have twice the thermal conductivity as diamond [4], [6], high stiffness, and high plasticity [4], [15], [16]. These

intriguing properties of carbon nanotubes enable many valuable applications in diverse fields, including medicine (disease diagnosis, drug delivery, cancer treatment, biosensors, biomedical imaging, and tissue engineering) [4], [7]–[12], electronics (beyond-silicon microprocessors, microelectronic components, and carbon nanotube field effect transistors) [1], [4], [6], chemistry (CNTs are good catalysts) [4], [17]–[19], energy storage (hydrogen fuel cells) [13], and many, many more.

The discovery of carbon nanotubes is commonly attributed to Sumio Iijima who in 1991 discovered multi-walled carbon nanotubes (MWCNTs) [20] and in 1993 discovered single-walled carbon nanotubes (SWCNTs) [21]. However, this material was in fact discovered nearly four decades earlier by Russian scientists who detected MWCNTs with a transmission electron microscope [22], [23]. Additionally, SWCNTs were independently discovered by Bethune, Kiang, Vries, *et al.* [24]. Although this paper was submitted after that of Sumio Iijima, both appeared in the very same issue of NATURE in 1993 [23]. Furthermore, CNT-like carbon filaments were created around the turn of the twentieth century [23].

Carbon nanotubes are among the family of carbon (C) isomers. Diamonds are well known for their aesthetic properties, but are in fact the 3-dimensional carbon isomer, have C-C single bonds, and are electrical insulators with a bandgap of $E_g = 5.47\,\text{eV}$. Graphene is a 2-dimensional sheet of carbon atoms with double bonds between them creating a hexagonal pattern. Graphene is the 2-dimensional carbon isomer and is a semimetal, a metal with a very small overleap between the conduction and valence bands. CNTs are often described as graphene sheets rolled into tubes. Thus, they are 1-dimensional structures still having double bonds. However, they are possibly the most interesting carbon isomer in that their properties change depending on how they are rolled. This changes whether they exhibit metallic or semiconducting properties, determines the bandgap, and specifies the optical absorption spectrum. Finally, fullerene is a sphere-like, 0-dimensional carbon structure that behaves as a semiconductor with a fixed bandgap of $E_g = 1.9\,\text{eV}$ [25, Table 1.1].

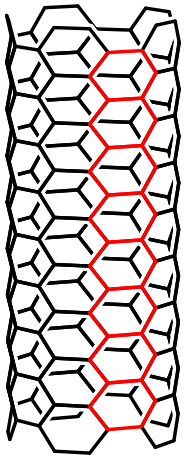
Unfortunately, these exciting features are not all unlocked by just any carbon nanotube. CNTs are designated by their number of walls and chirality; their electrical, mechanical, and optical properties depend heavily on both. CNTs are tube-shaped structures consisting of a single sheet or multiple sheets of graphene. The former case is referred to as single-walled (SWCNT), while the latter is multi-walled (MWCNT). In the second case, multiple concentric graphene tubes of different diameters are connected via van der Waals forces [4]. Carbon nanotubes are further classified by their chirality, the angle with which the graphene sheet is rolled onto itself. Imagine rolling a standard rectangular sheet of paper into a cylinder. The long edges can be brought together, creating a narrow tube, or the short

edges can be connected, creating a wide tube. If these are classified as rolling angles of 0° and 90°, respectively, then it is also possible to roll the paper with any angle between 0° and 90° with a corner sticking out at each end. Similarly, CNTs appear with rolling angles between 0° and 30°, which is explained by graphene having order 6 rotational symmetry (rotating a hexagon by one-sixth turn results in an identical hexagon) whereas rectangular sheets of paper have order 2 rotational symmetry. For carbon nanotubes, a rolling angle of 0° creates a so-called "achiral zigzag" tube where each hexagon has two edges that are *parallel* to the central axis. Equivalently, a rolling angle of 30° creates a so-called "achiral armchair" tube where each hexagon has two edges that are *perpendicular* to the central axis. These are so named because of the pattern of carbon bonds at their extremities, which can be seen in Figure 2.1 and by the red annotations of Figure 2.2a. Any angle in between results in a "chiral" CNT. This process is symmetrical around 0°. That is, rolling angles up to −30° are possible and represent the enantiomer (mirror image). Often, such mirrors are referred to as having the same helicity with opposite handedness, while chirality fully defines both [26]. Furthermore, since graphene has order 6 rotational symmetry, the angles outside of −30° to 30° are redundant. For example, the range 60° to 90° is identical to 0° to 30°.

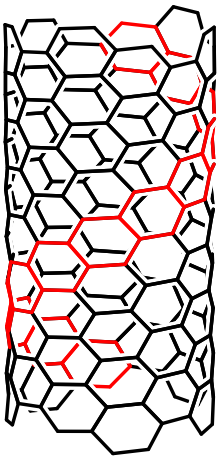
Unlike the paper-rolling example, this angle can only take discrete values for CNTs. Graphene is comprised of discrete carbon atoms, so the chirality vector (n, m) must land directly on a cell of the hexagonal grid presented in Figure 2.2. Also note that in reality, carbon nanotubes are not fabricated by rolling graphene sheets. This analogy is simply a mental aid to visualize the structure and chirality vector. Furthermore, the chirality is not "chosen" while manufacturing CNTs. As will be presented later in this section, modern carbon nanotube synthesis techniques are unable to select for chirality, which is the *raison d'être* of this project.

The diameter of a CNT is determined by the chirality, specifically by the length of the chirality vector. Like the paper example, the rolling angle indeed affects the diameter of the CNT, but only because this angle changes the distance between cells on the hexagonal grid. The diameter is affected more dramatically by the number of carbon atoms on a ring around (the perimeter of a right section) the tube. This becomes difficult to define graphically for chiral CNTs, but for zigzag and armchair, this quantity is simply the number of cells crossed by the chirality vector.

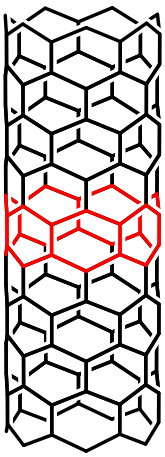
The chirality vector is a 2-dimensional vector (n, m) in a hexagonal grid that fully defines the chirality of a CNT. The two axes of this vector are not orthogonal as they would be for a traditional coordinate system, but rather are separated by 60° and perpendicular to two edges of each hexagon in the grid. Cells along these axes have the form $(n, 0)$ or $(0, m)$ and correspond to zigzag CNTs. The chirality vector $\vec{C_h}$ is defined more formally in



(a) Achiral Armchair SWCNT with chirality $(5, 5)$



(b) Chiral SWCNT with chirality $(7, 5)$



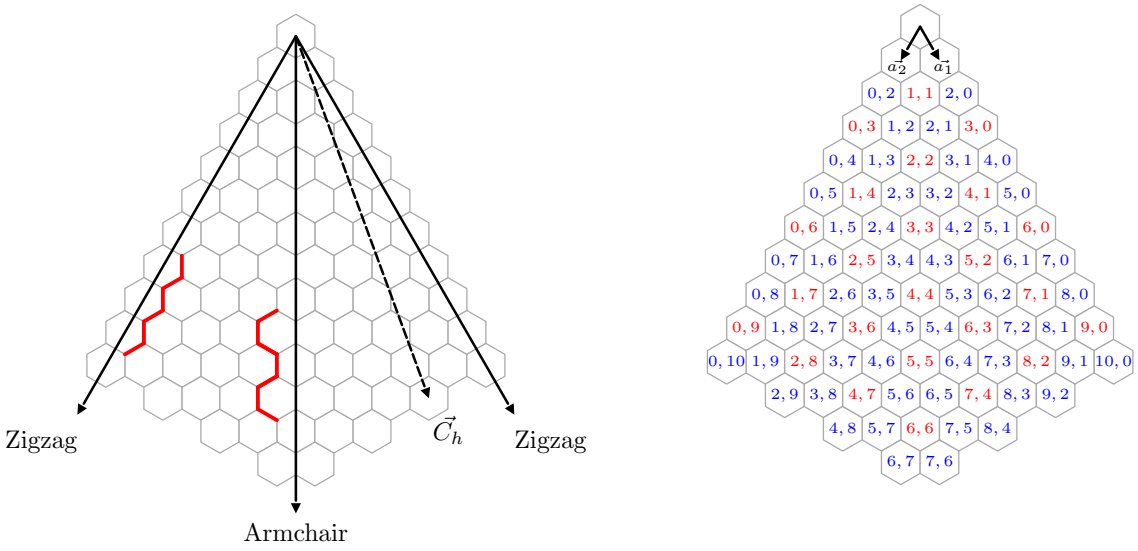
(c) Achiral Zigzag SWCNT with chirality $(7, 0)$

Figure 2.1: Visual representations of exemplary armchair, zigzag, and chiral SWCNTs.

Equation 2.1, where $\vec{a_1}$ and $\vec{a_2}$ are unit vectors defined in Figure 2.2b. Points (n, n) that have the same coordinate along each axis correspond to armchair CNTs, whereas points (n, m) where $m \neq n$ and $m \neq 0$ are chiral. The hexagonal grid and chirality vector are presented in Figure 2.2.

$$\vec{C_h} = n \cdot \vec{a_1} + m \cdot \vec{a_2} \equiv (n, m) \tag{2.1}$$

The chirality of a CNT defines many of its properties as briefly mentioned many times already. Indeed, SWCNTs and double-walled carbon nanotubes (DWCNTs) exist in metallic and semiconducting variants. These nanotubes are metallic if and only if $(n - m)$ mod $3 = 0$ (the difference between n and m is a multiple of three), and are semiconducting otherwise [25], [27], [28]. The metallic/semiconducting status of many chiralities is shown by the red and blue text colour of Figure 2.2b. An interesting consequence of this is that all armchair-type SWCNTs are metallic. Carbon nanotubes with three or more walls are always metallic [29]. A unique and fruitful property of semiconducting SWCNTs is that unlike traditional semiconducting materials, they can have different bandgaps, which is also directly determined by the chirality [26], [30]. This is advantageous in certain applications requiring specific bandgaps. Related to the bandgap is the absorption spectrum, which is also dependent on the chirality. This manifests as different visible colours for CNTs of different chiralities [26], [30], [31] as seen in Figure 2.3.



(a) The chirality vectors producing left- and right-hand zigzag, armchair, and chiral $(9, 2)$ SWCNTs.

(b) The unit vectors $\vec{a_1}$ and $\vec{a_2}$ and the chiralities of each cell. Metallic chiralities are marked with red text.

Figure 2.2: Visualisation of exemplary chirality vectors, unit vectors, and chiralities on a hexagonal grid representing the bonds of a graphene sheet.

2.2 Methods of Carbon Nanotube Synthesis

Multiple techniques have been devised over the years for carbon nanotube (CNT) synthesis, which are detailed over the following sections. Regrettably, these methods all lack selection for the chirality in the synthesised CNTs. That is, the resulting solutions are not purified having a single chirality, but rather have a mixture of multiple chiralities. This is unfortunate, as many experiments require single-chirality CNT solutions. It is however possible to sort carbon nanotubes after they are synthesised, which is the purpose of this thesis. The sorting of CNTs is discussed in Section 2.3.

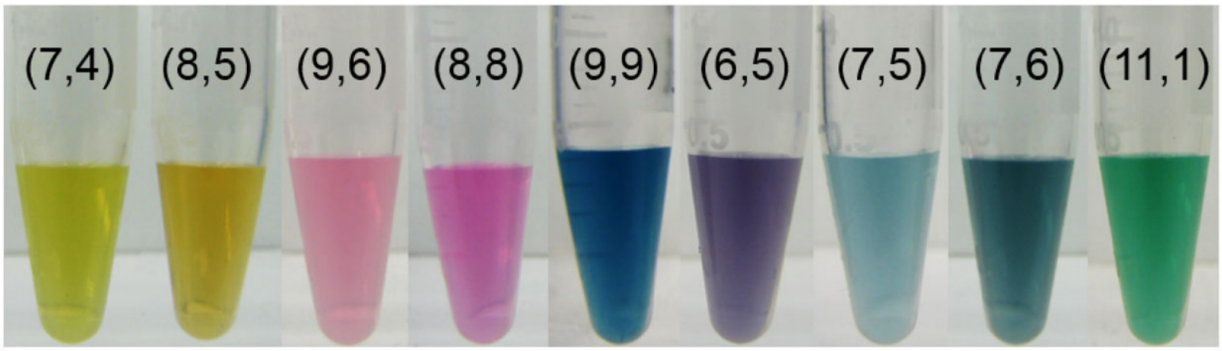


Figure 2.3: A photograph of many purified single-chirality carbon nanotube samples showing the relationship between colour and chirality. Reproduced from Ao, Streit, Fagan, *et al.* [31] with permission.

2.2.1 Electronic Arc Discharge

Electronic arc discharge is the oldest method of CNT synthesis [4], [32]. This method was initially used to produce carbon fibres [33], was then adapted to synthesise fullerenes [34], and finally was used by Sumio Iijima to produce MWCNTs [20] and SWCNTs [21]. This procedure has since been reproduced in hundreds of publications [32].

The principle of this method is to use arc discharge, the electrical breakdown of a gas to generate a plasma, in order to synthesise CNTs from a carbon precursor. A chamber such as the one presented in Figure 2.4 is used to contain the procedure. Inside are two electrodes: the anode, into which the powdered carbon precursor is filled, and the cathode, typically a graphite rod, onto which the CNTs will be formed. A voltage of around 15 V to 30 V is applied across the electrodes, creating an extremely high temperature plasma that sublimates the carbon precursor in the anode into a carbon vapour. This vapour aggregates on the cathode, resulting in CNTs and carbon soot. This process is carried out while the chamber is filled with an inert gas at low pressure [32].

A powdered metal catalyst such as nickel, iron, or cobalt is typically included in the anode with the carbon precursor. This catalyst can prioritise the formation of SWCNTs rather than MWCNTs and helps to bring the synthesised particles to the walls of the chamber rather than to the cathode [32].

This early method of CNT synthesis has notable drawbacks. First of all, the process is very sensitive to many factors, specifically the type of inert gas, the temperature, the catalyst, and the electric current. A fluctuating arc causes unstable plasma, which significantly degrades the CNT quality. Thus, the current must be carefully controlled to guarantee quality, yield, and nanotube size. Second, maintaining such an arc consumes enormous amounts of energy. Third, this process relies on carbon vaporisation, which does not scale well and gives ample opportunity for contamination. The resulting material must be purified, which is costly [35].

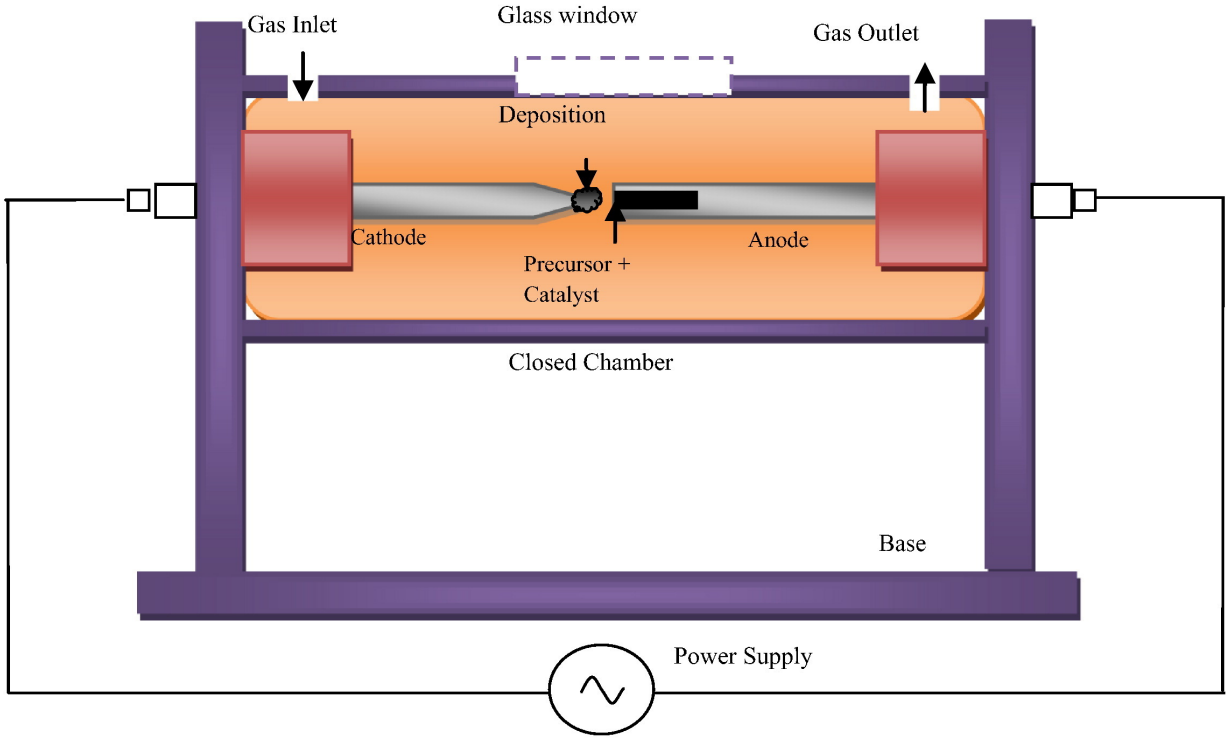


Figure 2.4: Schematic of an electric arc discharge chamber. Reproduced with permission from Arora and Sharma [32]

2.2.2 Laser Ablation

Laser ablation is very similar in many ways to electronic arc discharge. That is, both use a graphite precursor and typically a catalyst, and both are performed in an enclosure filled with an inert gas. The difference is that in laser ablation, the carbon is vaporised by direct contact with a powerful laser beam rather than a plasma. The resulting CNTs are carried by the gas to a collector. The process is typically carried out at approximately 1200 °C [36], [37]. An illustration of this process is presented in Figure 2.5.

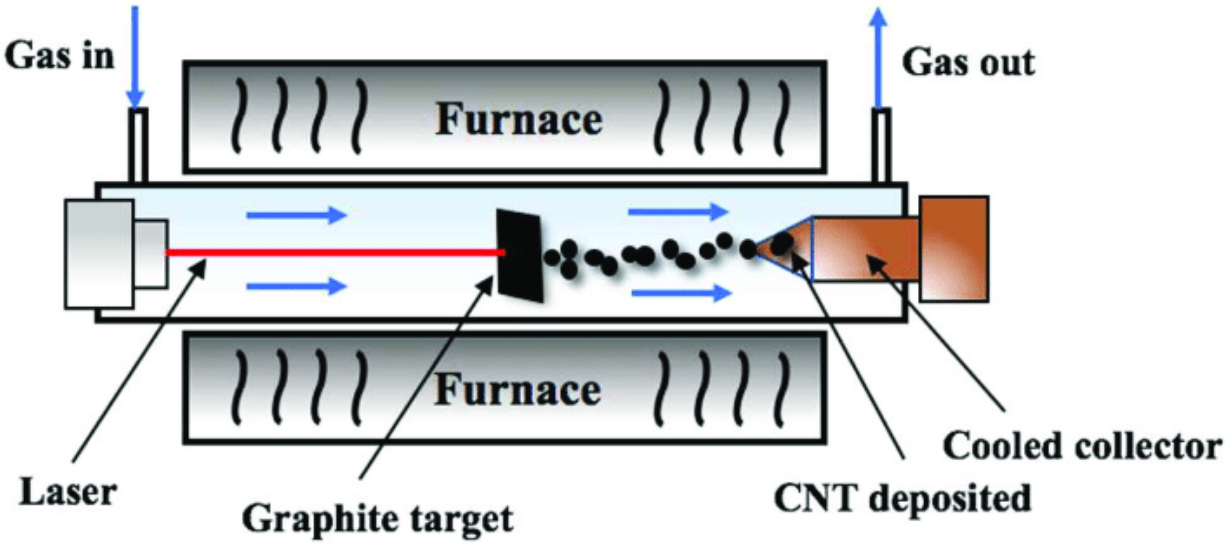


Figure 2.5: Schematic of a laser ablation CNT synthesis process. Reproduced unmodified from Lu, Raad, Safaei, *et al.* [38] under the CC BY 4.0 license.

The advantages of this method over electronic arc discharge are that it enables greater control over the growth conditions and produces carbon nanotubes of better quality with a high yield [36]. However, since this method is very similar to electric arc discharge, it has all of the same disadvantages. It is inefficient in its energy use, does not scale well, and invites contamination [4].

2.2.3 Chemical Vapour Deposition

An affordable and widely used alternative to the previous two methods is chemical vapour deposition (CVD). In this method, hydrocarbons are decomposed by high temperatures between 500 °C–1200 °C to synthesise carbon nanotubes on a catalyst consisting of metallic powder on a substrate. The metal used is typically iron, nickel, or cobalt and the support is silicon dioxide, ceramic, quartz, or glass. The diameter of the nanotubes is determined by the size of the metal particles [4], [39]–[41].

The highest known yield of CNT synthesis used chemical vapour deposition; in 2008, Kumar and Ando obtained 6 g of CNTs after only 30 minutes of CVD [39]. Unlike the previously discussed physical methods where the carbon precursor comes in direct contact with plasma or a laser beam, this method follows a chemical process. Contrary to other

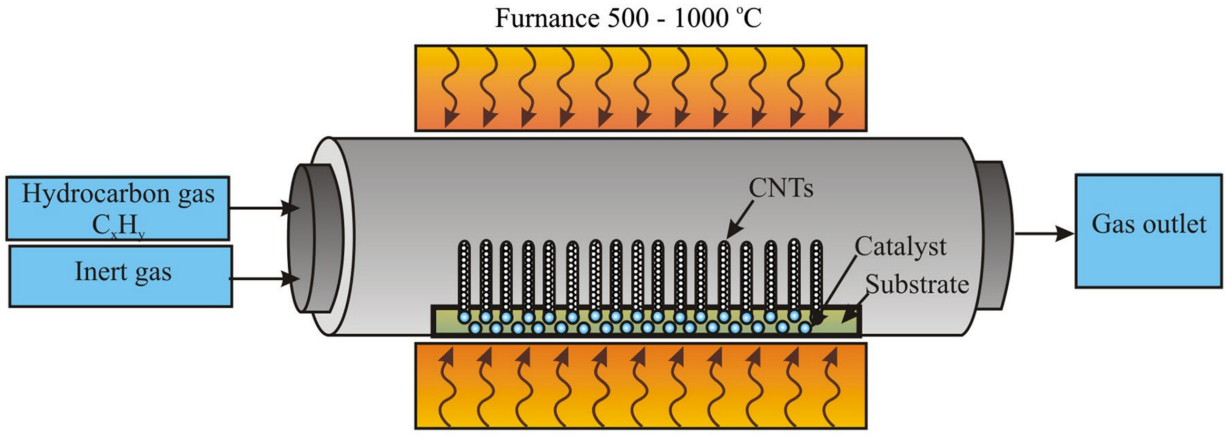


Figure 2.6: Illustration of the chemical vapour deposition process. Reproduced unmodified from Zaytseva and Neumann [42] under the CC BY 4.0 license.

methods, the raw materials are abundantly available and the energy requirements are greatly reduced. The process is performed continuously rather than by batch, and the process control is easy and can be automated. Furthermore, variations on this method like plasma enhanced CVD are able to produce vertically aligned nanotubes [43]. Thus, chemical vapour deposition is a viable option for affordable mass production of CNTs.

2.2.4 Gas Phase Chemical Vapour Deposition

The methodologies of high pressure carbon monoxide (HiPco®) and cobalt–molybdenum catalyst (CoMoCAT®) build on top of chemical vapour deposition with a gas phase catalytic process. The novelty with this technique is that high pressure (1 atm to 10 atm) carbon monoxide is used to supply carbon to the system in place of a solid precursor and iron pentacarbonyl ($Fe(CO)_5$) is used as the catalyst [44]. Bronikowski, Willis, Colbert, *et al.* were able to achieve a large scale production of 10 g of SWCNTs per day using the HiPco® method[45]. An image of such a high pressure carbon monoxide setup is presented in Figure 2.7.

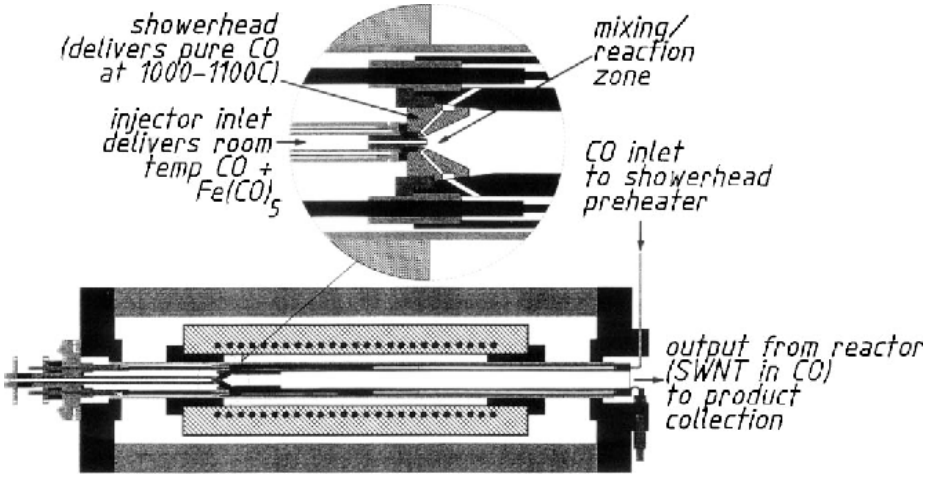


Figure 2.7: Illustration of the HiPco® SWCNT synthesis procedure. Reproduced with permission from Bronikowski, Willis, Colbert, *et al.* [45]. Copyright 2001 American Vacuum Society.

The CoMoCAT® method, on the other hand, is designed to be highly selective, inhibiting the creation of undesired forms of carbon and even limiting the set of chiralities produced simplifying the sorting step. This method exploits carbon monoxide (CO) disproportionation, which is the decomposition of CO into C and carbon dioxide (CO_2). This process takes place at 700 °C to 950 °C in pure CO at a total pressure around 1 atm to 10 atm. The ingenious trick that enables the performance of the catalyst is a careful mixture of cobalt and molybdenum. The synthesis is inactive if the mixture has too little cobalt, and unselective if too little molybdenum [46]. A schematic of the CoMoCAT® process is presented in Figure 2.8.

The clear advantage of this method over its counterparts is the high selectivity. Not only does it inhibit unwanted forms of carbon, but it is also highly selective with the diameter of the SWCNTs synthesised. Yet, it is not possible to produce pure single-chirality SWCNTs, so sorting is still required. The process requires less severe conditions than alternatives, and is scalable and affordable [46].

2.3 Methods of Carbon Nanotube Sorting

The introduction of this chapter justified the desire for CNTs with a single chirality. Many of the applications of these nanoscopic particles like the varying bandgap or metallic/semi-

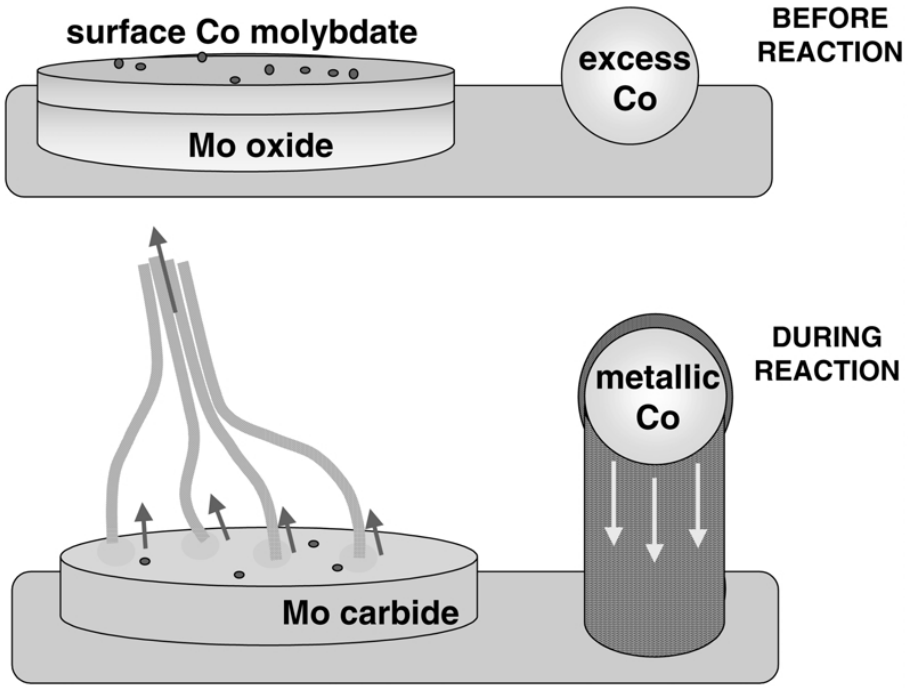


Figure 2.8: Illustration of the CoMoCAT® SWCNT synthesis procedure. Reproduced with permission from Resasco, Alvarez, Pompeo, *et al.* [46].

conducting behaviour are unlocked only if the chirality can be selected. However, the review of synthesis methods from Section 2.2 showed that, while some procedures like CoMoCAT® are more selective than others, none of the methods for CNT synthesis that exist today are able to fully select a single nanotube diameter. Thus, sorting techniques able to separate carbon nanotubes by their metallic/semiconducting status and chirality are required. The problem is defined more rigorously as extracting and isolating each and every single chirality given a mixture of SWCNTs species [26].

Many such techniques have been attempted over the past decades. These are largely adopted from biochemistry, as this field deals with separation problems routinely and because biomolecules like deoxyribonucleic acid (DNA) are exceptional at manipulating molecules in water. The most notable alternative sorting techniques are briefly described in Section 2.3.1, then aqueous two-phase extraction, the sorting method used in this project, is detailed in Section 2.3.2.

2.3.1 Historical Sorting Processes

The first successful separation of SWCNTs occurred in 2003 using the DNA-based ion exchange chromatography method [47], [48]. This is an atomic-structure-based (as opposed to bandgap-based) sorting method that separates DNA-wrapped SWCNTs based on their differential absorption and desorption onto resins. This early method did not produce impressive results for today's standards, but was invaluable to pave the way for future techniques and to show that SWCNT sorting was possible.

The density gradient ultracentrifugation method uses the intuitive principle that different chiralities of carbon nanotubes will have different densities. Thus, ultracentrifugation can be used to force the denser structures down while lifting the less dense structures to the top, as was shown by Arnold, Stupp, and Hersam in 2005 [49]. Around the same time, Wenseleers, Vlasov, Goovaerts, *et al.* attempted to separate CNTs by dispersing them in surfactants (sodium cholate (SC) and sodium deoxycholate (DOC)) before performing density gradient ultracentrifugation [50]. This led to the eventual momentous separation of single-chirality semiconducting SWCNTs [51].

This method is attractive in concept. CNTs with different diameters have different densities and so are separated by centripetal force. In practise, however, the buoyant density is not only dependent on the chirality, but also on the surface coating, hydration layer, and counter-ion distribution, limiting the effectiveness [26].

Selective extraction marked another breakthrough in CNT sorting. In 2007, Nish, Hwang, Doig, *et al.* discovered that polyfluorenes could be used to achieve selective dispersion of semiconducting SWCNTs in organic solvents [52]. This technique was quickly reproduced and expanded. Mistry, Larsen, and Blackburn were able to perform large-diameter extraction using a derivative of polyfluorene [53]. Furthermore, it was found that aromatic molecules can replace the previously used conjugated polymers [54], [55], which enables the possibility of using other aromatic molecules for this extraction [26].

Gel chromatography, which is similar to ion exchange chromatography, allows extraction of semiconducting tubes and chirality sorting. In 2009, Tanaka, Jin, Miyata, *et al.* discovered that semiconducting SWCNTs are more strongly attracted to polysaccharide gels than metallic SWCNTs, allowing the former to be extracted [56]. Two years later, Liu, Nishide, Tanaka, *et al.* found that the hydrogel to CNT binding can be influenced by chirality, not only metallic / semiconductor behaviour, by carefully choosing the surfactant composition [57].

2.3.2 Aqueous Two-phase Extraction

The final CNT sorting method that will be discussed, and the method chosen for this project, is aqueous two-phase extraction (ATPE). The first documented aqueous two-phase system was reported over a century ago when in 1896, Martinus Willem Beijerinck accidentally found that an aqueous starch solution was immiscible with an aqueous gelatin solution [58]. This was later exploited by Albertsson who, a century later in 1970, used a two-phase system composed of polyethylene glycol (PEG) and dextran (DX) in order to purify chloroplasts [59]. In the intervening years, aqueous two-phase systems have had numerous applications in biochemistry, enabling the separation and purification of biomolecules, proteins, enzymes, membranes, monoclonal antibodies, DNA, and viruses [60]. In 2013, aqueous two-phase extraction (ATPE) was adapted for the sorting of CNTs by Khripin, Fagan, and Zheng [61]. This has been reproduced and expanded upon in many subsequent publications [30], [62]–[67]. The use of ATPE for CNT sorting has enabled bulk purification (removal of carbon soot and other undesirable forms of carbon), robust separation based on chirality and semiconducting/metallic status for a wide range of diameters, and even the sorting by handedness and helicity of metals, quasi-metals, and semiconductors via DNA-wrapped CNTs [26], [31], [68]. This is a breakthrough for CNT sorting.

The method is called "aqueous" because each phase is primarily water. Despite the similar water content, the two phases can still be separated due to the relative density of the two polymers used. A two-phase system is achieved under ambient conditions via gravity given enough time, or can be achieved quickly by low-speed centrifugation. Separation of metallic and semiconducting CNTs is also possible via the reduction-oxidation process [62]. Once two phases are formed, they can be separated by careful pipetting or decantation. The process can be repeated on the resulting two subpopulations, resulting in highly purified CNTs after sufficient iterations.

ATPE is similar to the previously discussed methods in that SWCNTs are wrapped by surfactants or DNA and are separated by centrifugation based on their density and hydrophobicity. However, ATPE is superior to these methods in that the two phases are homogeneous, improving resolution, allowing multiple consecutive separation steps, and providing reproducible results even with varying surfactant concentrations. In this way, it is possible to obtain highly purified CNTs by applying multiple stages of ATPE as presented in Figure 2.9. Simultaneous separation based on handedness and helicity is only possible with these improvements. Additionally, because of the ingredients used and moderate equipment requirements, ATPE is much more scalable and affordable than alternatives. Because of the polymers and surfactants used, this procedure is environmentally-friendly [60]. All of these factors lend ATPE to mass production [26], [30].

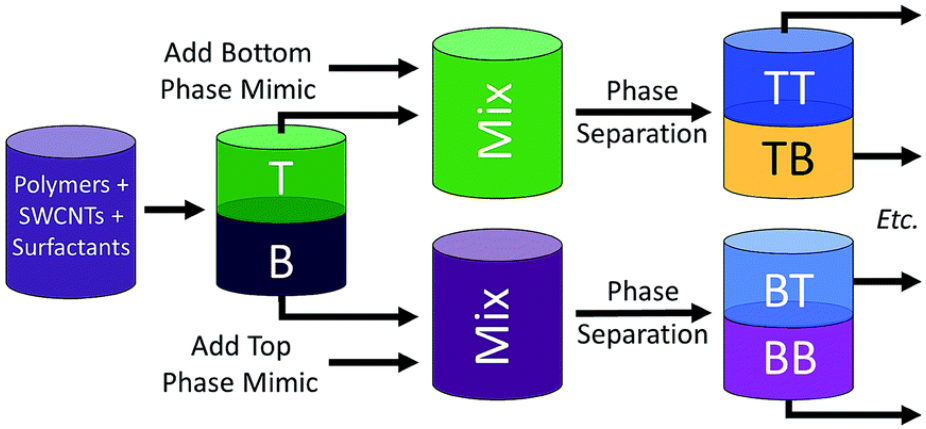


Figure 2.9: Schematic of sequential ATPE separations, forming a binary tree. By applying many stages, a high purity can be achieved. After each stage, surfactants (Phase Mimic in the image) are added to control the next separation. The characters T and B are successively added to sample labels according to their path through the tree. Reproduced unmodified from Fagan [68] under the CC BY 3.0 license.

Two factors are crucial to successful aqueous two-phase sorting. First, great care must be taken with respect to the concentration of the two polymers making up the polymer-polymer two-phase system. As shown by the phase diagram of Figure 2.10, an incorrect polymer mixture will result in a single phase, which is obviously undesirable for two-phase extraction. A common choice for the two polymers in biology and CNT sorting alike is polyethylene glycol (PEG) and dextran (DX) [26], [30], [60], [68]. PEG has a mass of 6 kDa while DX has a mass of 70 kDa, meaning that DX partitions to the bottom phase and PEG partitions to the top phase during centrifugation.

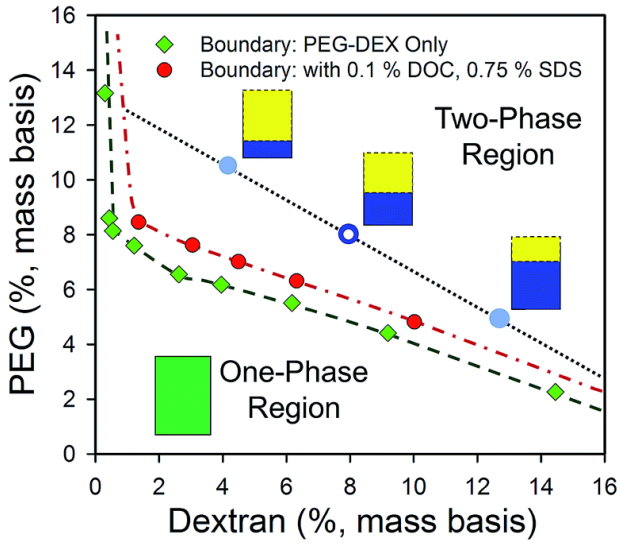
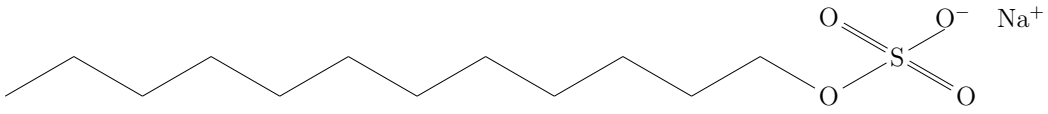


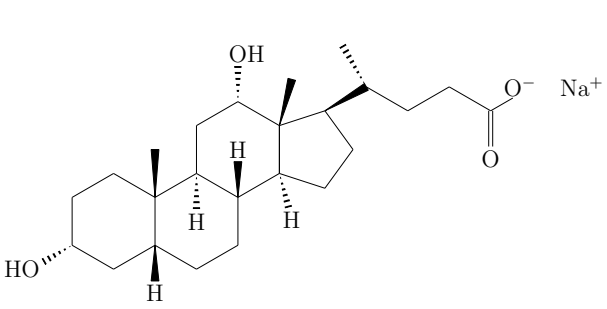
Figure 2.10: An exemplary polymer-polymer phase diagram with and without surfactants for 6 kDa PEG and 70 kDa DX. PEG-DX mixtures below the curve will result in a single phase, which is undesirable. Mixtures above the curve will result in two phases. Reproduced unmodified from Fagan [68] under the CC BY 3.0 license.

Second, the surfactant concentration and type must be carefully selected depending on the CNT diameter and separation conditions. Indeed, sodium dodecyl sulfate (SDS) and sodium deoxycholate (DOC) are used for diameter separation, while sodium dodecyl sulfate (SDS), sodium cholate (SC), and sodium hypochlorite (NaClO) are used for metallic/semiconducting separation. As was briefly mentioned previously, surfactants as well as polymers are required for CNT ATPE extraction. These attach themselves onto the nanotubes based on their diameter or electrical properties. This process can be understood intuitively for diameter separation after observing the chemical structures of the surfactants involved as presented in Figure 2.11. The long tail of SDS wraps around large-diameter CNTs, while DOC can attach itself to CNTs of any diameter. Thus, some amount of SDS is required to extract large-diameter CNTs as can be seen in Figure 2.13.

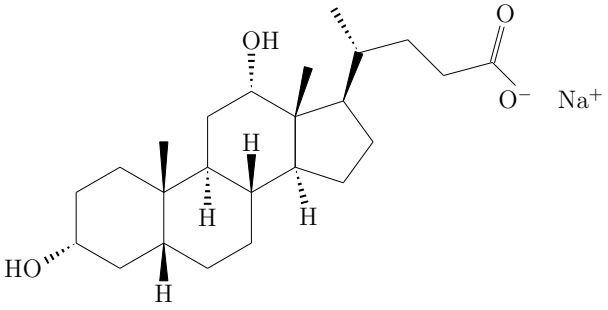
These surfactants are drawn to either PEG or DX, whichever has a more similar hydrophobicity [65], [71]. This partitioning of the surfactants brings the CNTs along with them to the appropriate phase, sorting the CNTs. Specifically, SDS is partitioned to the top phase while SC and DOC are partitioned to the bottom phase. It is also possible to use DNA in place of surfactants [26], [31], [61]. However, in this project, surfactants were chosen. The concentrations of the surfactants are selected based on the desired diameter



(a) Sodium dodecyl sulfate (SDS)



(b) Sodium cholate (SC)



(c) Sodium deoxycholate (DOC)

Figure 2.11: Chemical structure of the surfactants used for ATPE. The figures are original, although chemical structures are from Arnold, Green, Hulvat, *et al.* [69] and Green, Duch, and Hersam [70].

with which to divide the CNTs.

Typically when selecting the concentrations for the surfactants, the amount of SC or DOC is fixed, and the amount of SDS is selected accordingly. The diagrams in Figure 2.12 show how the SDS concentration can be selected for a given concentration of DOC ((a), chirality separation) or SC ((b), metallic/semiconducting separation). Note that these diagrams are general and without values; it is included only to convey the concept. In principle, the CNTs to the left of the chosen concentration will partition to the top phase and those to the right will partition to the bottom phase. This is illustrated in Figure 2.13, which shows how particles are partitioned depending on the concentrations of SDS and DOC. The SWCNT diameter extracted to the top phase as a function of SDS concentration for a specific ATPE system is presented in Figure 2.14.

Here, it is important to consider the partition coefficient: K_i. The partitioning of a CNT species to one phase or the other will never be perfect, as illustrated by the diagonal lines of Figure 2.14. There will always be some stray CNTs in the opposite phase due to thermodynamic equilibrium, which requires chemical potentials of nanotubes in both phases must be equal [61]. The partition coefficient K_i measures the ratio between the concentration of some CNT species i in the top and bottom phases, noted $c_{T,i}$ and $c_{B,i}$, respectively. The formula to calculate K_i is given in Equation 2.2, where $\mu^0_{T,i}$ and $\mu^0_{B,i}$ are the standard chemical potentials of i in the top and bottom phases, respectively, k_B is the

Boltzmann constant, and T is the absolute temperature [61], [68]. This has the benefit that the partition conditions can conveniently be tuned by nanotube length, surfactant concentrations, and temperature [61], [68].

$$K_i = \frac{c_{T,i}}{c_{B,i}} = \exp\left(-\frac{\mu^0_{T,i} - \mu^0_{B,i}}{k_B T}\right) \tag{2.2}$$

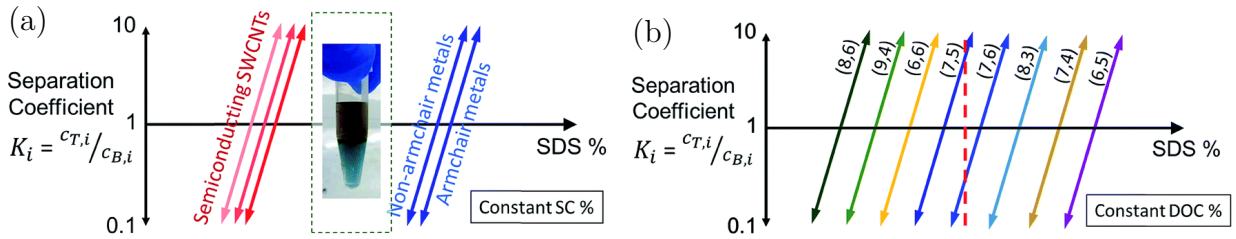


Figure 2.12: Partition coefficient diagrams showing how to select the concentration of SDS given (a) a fixed SC concentration for metallic/semiconducting separation and (b) a fixed DOC concentration for chirality separation. Reproduced unmodified from Fagan [68] under the CC BY 3.0 license.

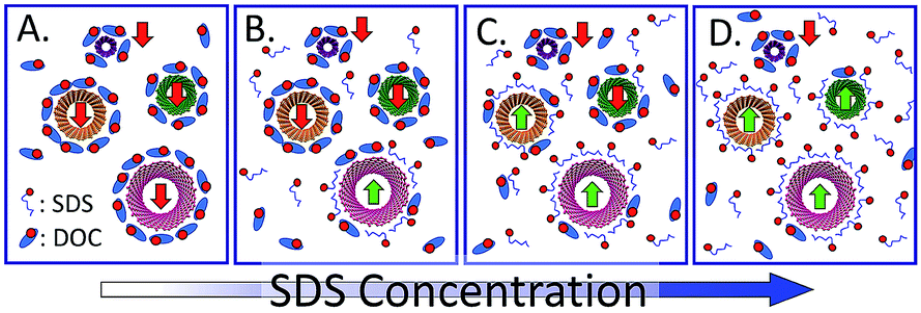


Figure 2.13: Simplified illustration of the effect of the SDS concentration on the partitioning of the two phases. Inscribed arrows show the partitioning direction. (a) For a low SDS concentration, all SWCNTs are covered with DOC and are partitioned downwards. (b) and (c) As SDS concentration increases, SDS competes with DOC and larger diameter SWCNTs are partitioned upwards. (d) At high SDS concentrations, almost all CNTs are covered with SDS and are partitioned upwards. Reproduced unmodified from Fagan [68] under the CC BY 3.0 license.

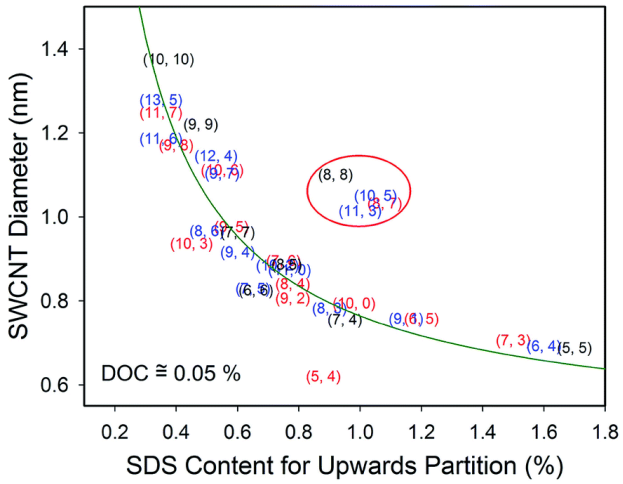


Figure 2.14: Exemplary curve showing the approximate SDS concentration required to partition the SWCNT with the annotated chirality into the top phase of a PEG 6 kDa:DX 70 aqueous two-phase system with the concentration of DX fixed at 0.05 %. Reproduced unmodified from Fagan [68] under the CC BY 3.0 license.

2.3.2.1 Pre-processing Steps

Several pre-processing steps are required before the main ATPE process can be performed. The carbon nanotubes must be dispersed, the impurities must be removed via centrifugation, and the stocks must be prepared in large volumes.

§1 Dispersing SWCNTs in surfactants The ATPE process, as indicated by "aqueous" in the name, is liquid based. However, CNTs are hydrophobic, meaning they cannot easily be dispersed in water. Hence, there is a need to disperse the nanotubes in surfactants, creating homogeneous solutions that can be used in a liquid-based protocol. Typically, DOC is used for SWCNTs and SC is used for MWCNTs. Since this project focuses on SWCNTs, DOC with a concentration of 1.6 % is used, which is the optimal value [30], [72]. Since CNTs are nanoscopic particles that are hazardous if inhaled, they are put into solution inside a glove box. Thus, this pre-processing step is not a target for automation.

After adding CNTs to DOC, the particles must be properly dispersed via sonication. Without this step, CNTs are clumped together due to high inter-particle van der Waals forces.

§2 Purification When purchasing synthesised CNTs, the powder typically contains approximately 95 % amorphous carbon, carbon soot, and leftover catalysts used from the synthesis process for the sources used in this project [73]–[77]. The CNTs are easily extracted via ultracentrifugation, for which a *Beckman Coulter Optima XE-100* [78] is used.

§3 Preparing Polymer and Surfactant Stocks The main ATPE process uses many chemicals as previously discussed. Two polymers (PEG and DX) as well as three surfactants (SDS, DOC, SC) are used. These chemicals are sold in a powdered form, so it is necessary to disperse them in deionized water for the liquid-based ATPE protocol. To increase the efficiency of the main protocol, it is advantageous to produce these stocks in bulk as a pre-process step. The concentrations for each chemical, based on the recommendation of Fagan [68], are presented in Table 2.1. Polymer concentrations are in the form mass per mass (% m/m) while surfactants are in the form mass per volume (% m/v).

Table 2.1: List of stocks showing chemicals and target concentrations. PEG is duplicated, meaning that both concentrations are required for the ATPE process. The volumes and masses are from HeeBong Yang and the original target concentrations are from Fagan [68].

Chemical Name	Target Concentration	Chemical Mass	Water Volume
DX	20 % (m/m)	10 g	40 mL
PEG	25 % (m/m)	12 g	36 mL
PEG	40 % (m/m)	20 g	30 mL
SDS	4.5 % (v/m)	2.025 g	40 mL
DOC	4.0 % (v/m)	1.6 g	40 mL
SC	4.5 % (v/m)	2.025 g	40 mL

The powdered chemicals must be accurately weighed and mixed in *Corning*® 50 mL polypropylene v-bottom tubes [79] with the appropriate amount of deionized water. In order to fully disperse the particles, the stocks then undergo bath sonication. The specifics of the automation of this process are elaborated in Section 4.4.

2.3.2.2 Main ATPE Process

The theory of aqueous two-phase extraction has been presented in Section 2.3.2. This section enumerates the steps required for this lab's implementation of ATPE specifically.

The main ATPE process takes place with a DOC concentration of 0.04 % for diameter separation and < 0.02 % for metallic/semiconducting separation [30], [68]. Since a higher concentration of 1.6 % was used in the pre-processing CNT dispersion step (Section 2.3.2.1 §1), the DOC concentration must be lowered. As discussed previously in Section 2.3.2, diameter separation uses SDS and DOC while metallic/semiconducting separation uses SDS and SC. With the DOC and SC concentration fixed depending on the separation type, the SDS concentration is varied according to the desired partitioning point (Figure 2.12).

Another complication to keep in mind is that the sorting method depends on the synthesis process of the CNTs. For example, SG65i-type SWCNTs produced by *NanoIntegris* [80] must start with diameter separation whereas SG76-type SWCNTs produced by *Sigma-Aldrich* [81] must start with metallic/semiconducting separation. This is because each process has a different bias towards producing different chiralities [30], [68], [82].

To accelerate the partitioning of the two phases, the *Thermo Scientific™ Sorvall™ ST 8* [83] low-speed centrifuge is used at a speed of 1500 RPM for 3 minutes. The main process is performed in 14.7 mL glass flat-bottom vials, which will be pertinent for future sections on ATPE automation.

A typical recipe applicable for SWCNTs produced via the CoMoCAT® or HiPco® procedures is presented in Figure 2.15. This recipe first specifies a purification step to discard any unwanted carbon structures (Section 2.3.2.1 §2). Next, two diameter separations are performed, followed by two metallic/semiconducting separations. This ordering is convenient for SWCNTs synthesised in this method because the subpopulations resulting from the diameter separations can be further sorted by metallic/semiconducting extractions without interference or "cross-talk" [82]. Additional recipes are available in [82].

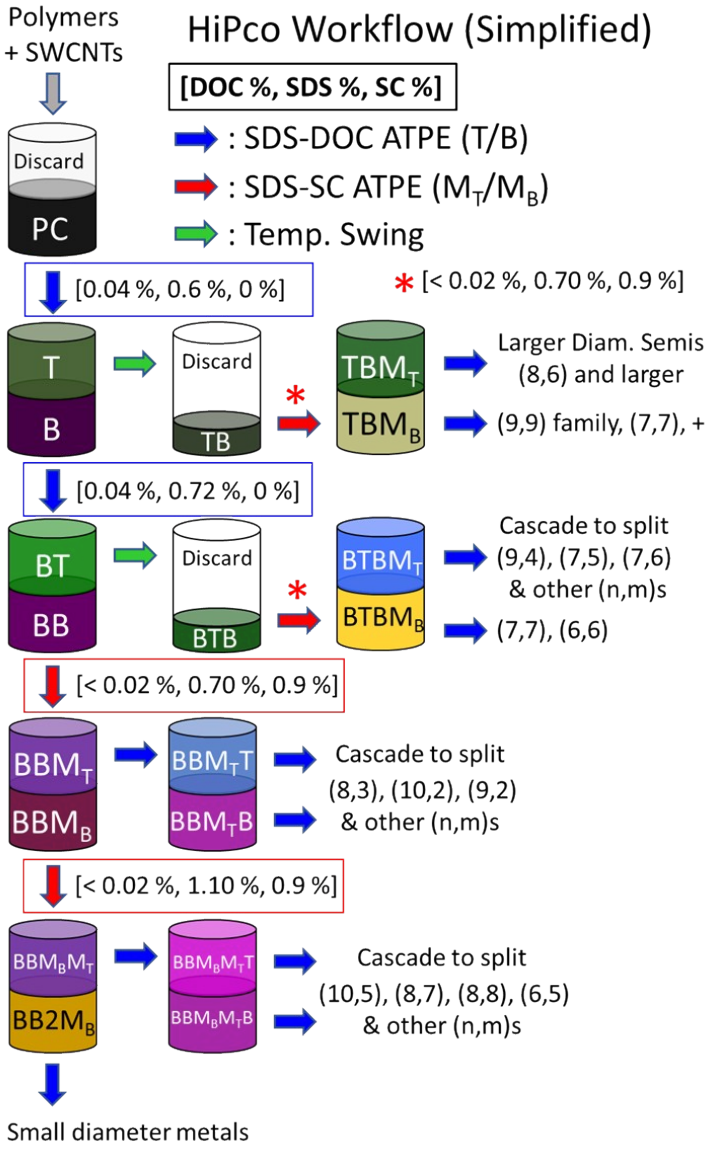


Figure 2.15: Schematic of a typical generalized ATPE recipe for SWCNTs synthesised by CoMoCAT® and HiPco®. Reproduced unmodified from Fagan [68] under the CC BY 3.0 license.

2.3.2.3 Post-processing Steps

There are two post-processing steps required following successful ATPE: characterisation and polymer exchange.

§1 Optical characterisation is crucial to ensure that the sorted SWCNTs have the expected chirality and purity. As discussed at the beginning of this chapter, SWCNTs have unique bandgaps defined by their chirality, which alters their absorption spectrum and colour. It is thus possible to validate the chirality of a sample using ultraviolet-visible (UV-Vis) spectroscopy. Examples of such spectra and how they can be used to identify SWCNT chiralities are presented in Figure 2.16. The spectrophotometer used for this project is the *VWR® UV-3100PC* [84]. The automation of optical characterisation is planned as a future work for this project, but it is not a high priority. Thus, it has been considered in certain preparation steps (for example, a place for the UV-Vis spectrophotometer was reserved on the automation table, see Section 5.1), but there are no short-term plans to begin this integration.

§2 Polymer exchange is the process of removing the chemicals added to the sample as required by ATPE. This is achieved by diluting the solution in DOC and filtering the liquid with a membrane, extracting polymers and lowering surfactant concentrations. This step is not in the current scope for automation.

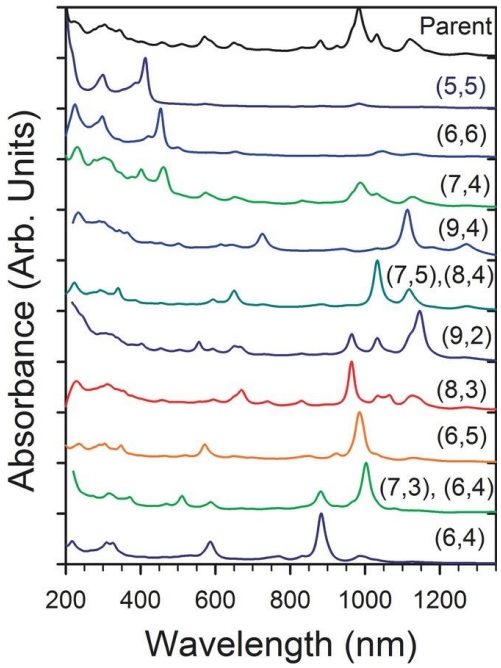


Figure 2.16: Absorption spectra of sorted SWCNTs with annotations showing the primary chirality in each. All fractions were derived from the parent population shown at the top. Reproduced with permission from Fagan, Khripin, Batista, *et al.* [64].

Chapter 3

Project Outline

3.1 Problem Definition

The goal of this project is to automate the sorting of SWCNTs by their chirality using the aqueous two-phase extraction (ATPE) procedure (Section 2.3.2). Although full automation is not yet achieved (Section 6.1), the ultimate objective is for the system to be fully autonomous such that it can be run unattended. The ATPE recipe should be programmed beforehand and will be executed from start to finish by the system. The purity and consistency of the sorted samples must be at least as good as when ATPE is performed manually, which can be verified by spectrophotometry. The automated system must have a throughput at least as high as manual ATPE, and must limit the waste of chemicals as much as possible. Finally, the system must perform the main ATPE process (Section 2.3.2.2) as well as stock preparation (Section 2.3.2.1 §3) and eventually characterisation (Section 2.3.2.3 §1).

3.2 Plan

This section details the plan established for the automation of ATPE. Section 3.2.1 justifies the choice of the most important component of the system: the liquid handling robot. The specifications of the selected robot are given in Section 3.2.2. As the *OT-2* lacks certain functionality required for ATPE, custom modules were designed for it, expanding its capabilities. These modules are presented in Chapter 4.

3.2.1 Liquid Handling Robot Selection

The most important aspect of ATPE is accurate small-volume pipetting. Thus, the first component selected and the component around which the rest of the system was constructed is the liquid handling robot. The role of this instrument is to perform all pipetting tasks in the ATPE process. It must be able to mix various stock liquids following a recipe to create the samples, to mix stocks to create mimics (pre-mixed commonly used stock mixtures), and to add the water to a powdered chemical to create a stock of the correct concentration. The selection of this machine is detailed in Section 3.2.2.

Several options were considered for the liquid handling robot, some providing only pipetting functionality and others offering advanced features such as powder handling as well. Turnkey options providing either base liquid handling functionality or all required functionality out of the box were considered in addition to custom implementations, where every element would be developed by the lab. Among the turnkey options are the *Opentrons OT-2* [85], the *Andrew Alliance Andrew+* pipetting robot [86], the *Anton Paar Modular Sample Processor* [87], and the *Sirius Automation MultiTasker®* [88], *MiniTasker®* [89], *MicroTasker®* [90], and *OmniTasker®* [91].

Ordinarily for such an important engineering decision, a decision matrix would be presented justifying the choice of an ideal option given a set of criteria. However, many of the proprietary options do not make public the details of their products, most notably the pricing. Thus, no decision matrix can be shown. Rest assured that the *Opentrons OT-2* was the clear winner. This option had two major advantages: its orders-of-magnitude lower cost and its commitment to openness. The open-source programming coming stock on the machine as well as the custom protocols informing it of what tasks to carry out are written in Python, making its programming extremely easy. It has been invaluable to have access to the open-source code of the robot, to have direct access to the developers via *GitHub* issues, and to participate in the software design through *GitHub* pull requests.

3.2.2 *OT-2* Liquid Handling Robot

The *Opentrons OT-2* is a liquid handling robot with a focus on accessibility and openness. The machine, which is presented in Figure 3.1, is similar in appearance and in function to a fused deposition modeling (FDM) 3D printer, an affordable kind of 3D printer popular in schools and libraries that builds a model layer by layer by extruding molten plastic through a nozzle. Like an FDM 3D printer, one or multiple instruments are moved in three dimensions. However, this instrument would be a nozzle for a 3D printer, while

the *OT-2* is equipped with up to two motorized pipettes. Multiple pipette capacities are available in single- and eight-channel variants. The so-called "deck" of the robot holds samples in vials, test tubes, or well plates and can also support many first-party modules, including a thermocycler, a magnetic module, and a temperature module [85]. The *OT-2* is highly configurable, as shown by Figure 3.2. The robot's operating system, control code [92], and even printed circuit boards are open source [93]. It was thus thought that it would also be easily expandable and adaptable to optimise for the specific use-case of ATPE.

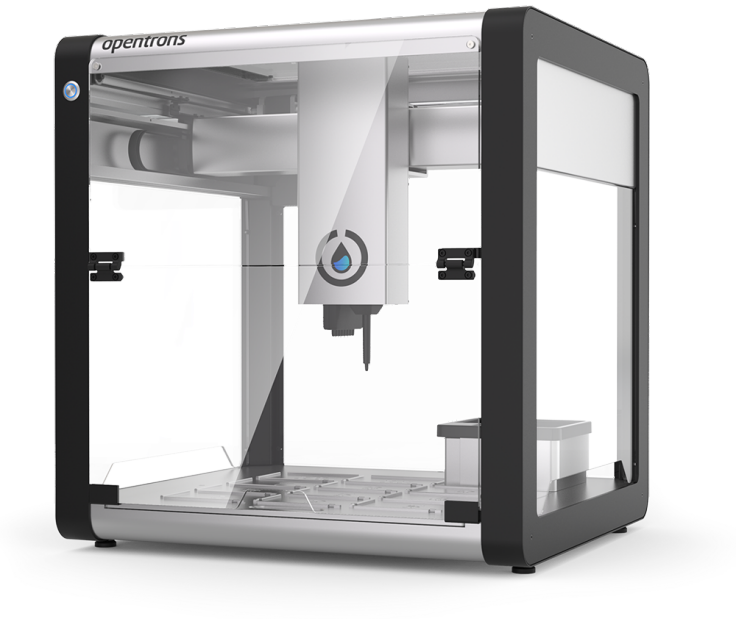


Figure 3.1: Image of the *Opentrons OT-2*. Reproduced from reference [94].

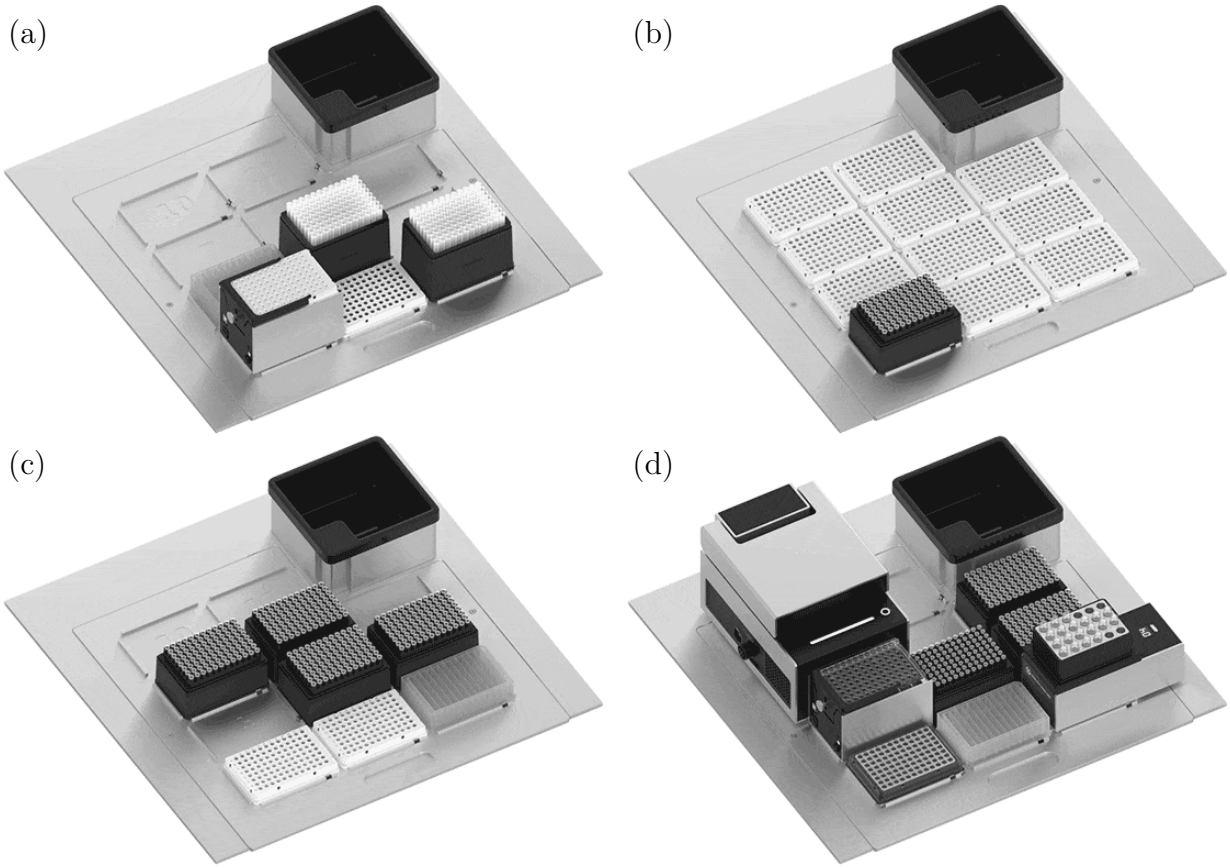


Figure 3.2: Four exemplary deck layouts showing multiple configurations of first-party elements including tip racks, well plates, and modules. Reproduced from reference [94].

Chapter 4

Custom Hardware and Software Components

Since this liquid handling robot is only capable of pipetting, additional external modules and augmentations to the *OT-2* are required to perform all other tasks mandated by ATPE. The first external module is the *Gripper*, whose task is translating containers (vials and tubes) inside the liquid handling robot as described in Section 4.1. Next, the *Vial Opener* was added, which is able to remove the cap of a container when paired with the *Gripper* as explained in Section 4.2. After this, the *Photo Stage* and associated image processing system was integrated. As Section 4.3.1 illustrates, this module is a platform to separate the top and bottom phases through the use of computer vision. The following module, which is explained in Section 4.4, is not part of the main ATPE process but rather aims to automate the prerequisite step of stock-making (Section 2.3.2.1 §3). Subsequently, the bar-coding system described in Section 4.5 introduces a mechanism capable to uniquely identify and track each vial and tube used by the ATPE process. Finally, the external robotic arm ties the *OT-2* into some larger external modules, closing the loop between this liquid handling robot and the centrifuge, sonicator, and spectrophotometer. Being a collection of off-the-shelf components rather than a custom module, this is detailed in Chapter 5.

The control centre that coordinates all of these distributed modules is the *Central Command* module, which is primarily a software module responsible for management and communication. The architecture follows a star pattern, with all commands and instructions going through this module. It is the task of this component to communicate with all modules, often in unique protocols, and pass messages between them. The reason for this architecture is that many of the off-the-shelf components use different protocols that

cannot be modified. Thus, it is impossible to use a common bus without adding a translator for each of these proprietary components. Therefore, the *Central Command* module serves the role of implementing each protocol only once, instead of implementing every protocol on every other peer. This architecture is also beneficial in its simplicity; it is easy to reason about and to diagnose issues. When a new module is added, the integration is straightforward; simply implement the module's interface in the *Central Command* module.

The software to enable this system of modules and instruments is explained in Section 4.6, including the firmware used by each module, the libraries created to facilitate the programming of Opentrons protocols, and custom Graphical user interfaces (GUIs), both standalone and extensions to the official Opentrons GUI.

The instruments included inside the *OT-2* are configurable using a deck with eleven identical slots. An image of this deck is shown in Figure 4.2. The space allocated inside the *OT-2* for each element, be it stock from *Opentrons* or custom, is presented in Figure 4.3. This plan includes stock tip racks for each pipette of the *OT-2*, stock vial racks for the 14.7 mL vials in which the main ATPE solutions are mixed, and the custom modules that are detailed in the following sections. Many of the custom modules are larger than a single deck slot or extend outside of the deck. The long *Vial Opener* (Section 4.2) and balance (Section 4.4.4) extend off the back of the deck, while the *Photo Stage* is elongated off the right edge so that the camera is as far as possible from the sample (Section 4.3.1). Finally, the large *Conveyor* of the *Powder Handling System* consumes three deck slots (Section 4.4). This layout is presented slightly out of order in that sizing was only determined and influential discoveries were only made while implementing the presented modules. For instance, the first row was filled with modules that do not interact with the *Gripper* (Section 4.1) — the tip rack for each pipette — because the *Gripper* cannot reach the first row due to its mounting position behind the pipettes.

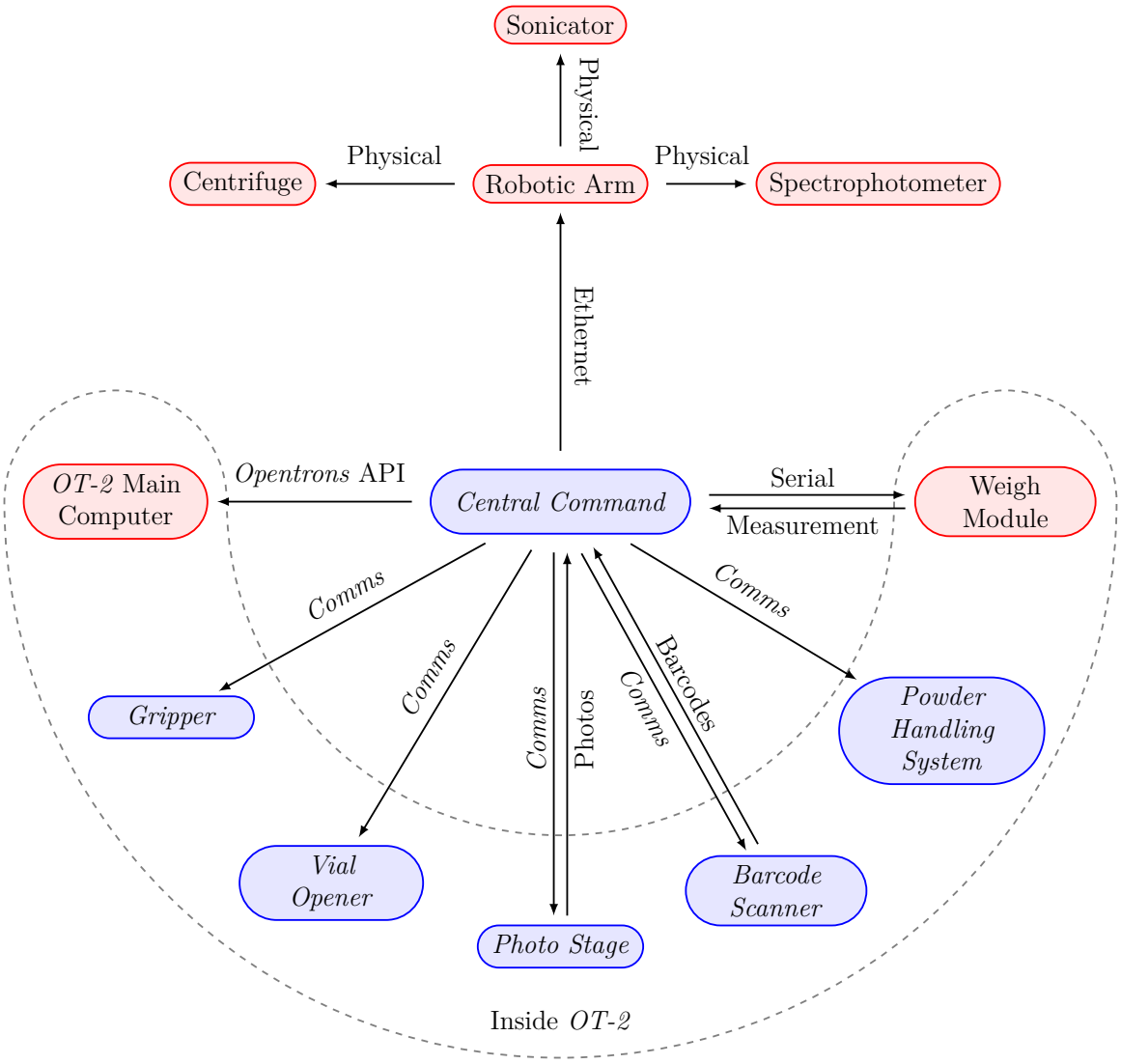


Figure 4.1: Representation of the star architecture communication between the modules of the automation system, with all messages passing through the *Central Command* module. Blue nodes are custom designs for this project and communicate on a standardised protocol (Section 4.6.1.1). Red nodes are third-party components using a proprietary communications protocol that the *Central Command* module must be adapted to understand. The gray dashed line indicates modules that are physically inside the *OT-2*. *Comms* refers to the standardised communications library (Section 4.6.1.1). The responses are omitted when uninteresting.



Figure 4.2: A 3D image of the *OT-2* deck, showing the slot numbering system and the ridges used for positioning. The slots are 128 mm wide and 86 mm deep [93]. The 3D model of the *OT-2* deck is provided by *Opentrons* [93], onto which the numbers were added.

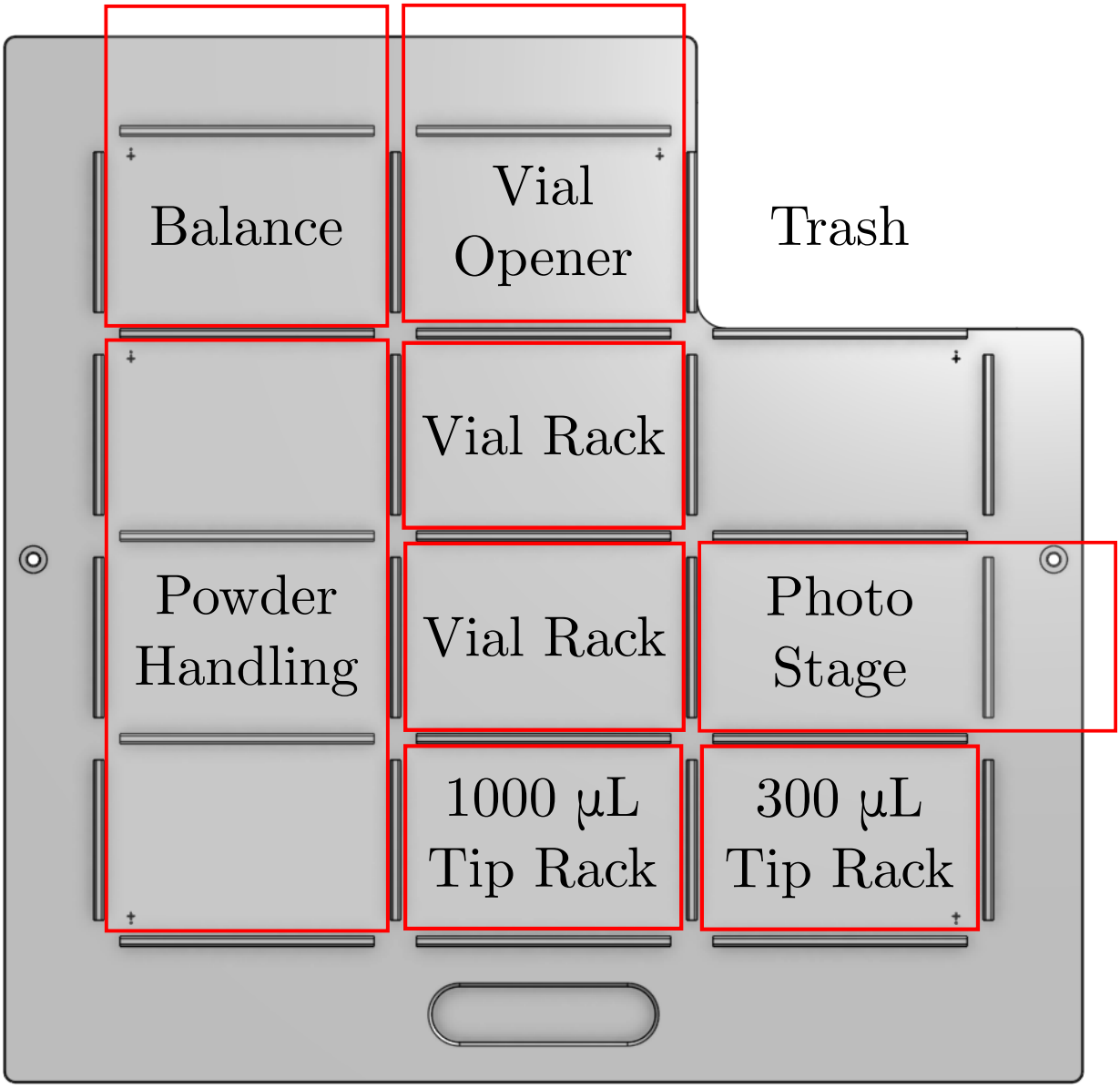


Figure 4.3: Planned layout of the modules on the deck of the *OT-2*. Many modules consume multiple deck slots or extend outside the deck. The deck slots are numbered from left to right, bottom to top, starting with 1 in the bottom left corner (Figure 4.2). The 3D model of the *OT-2* deck is provided by Opentrons [93].

4.1 *Gripper*

The *Opentrons OT-2* liquid handling robot is able to precisely transfer liquids between vials and tubes stored inside it. However, it is not equipped with any mechanism to transfer vials or tubes themselves. The *Gripper* is an addition to the *OT-2* enabling it to transfer vials between modules. In this way, it is possible to bring vials from racks to other custom modules, such as the *Vial Opener*, the *Weigh Module*, or the *Barcode Scanner*.

The first step when designing the *Gripper* was to define the constraints. It must be capable of lowering enough to take a vial out of any piece of labware (the *Opentrons* term for a tube rack, well plate, etc.) in the *OT-2*. To account for any future labware that might hold the *Gripper* lower than the standard vial rack, the *Gripper* was designed to lower sufficiently to take a vial directly off the deck of the robot. No module can bring the vial below the deck of the robot, that is physically impossible, so this constraint ensure that any future modules are compatible with the *Gripper*. The *Gripper* must also be capable of fully retracting, leaving clearance for the pipettes of the *OT-2* to operate. That is, the *Gripper*'s collapsed position must not leave any part below the tip of the pipette in the pipette's highest position.

The *Gripper* must be able to grasp and transport both the 50 mL tube and 14.7 mL vial. If this is not possible with a single gripper, two grippers can be used (one for each type of container) provided that together they satisfy all other constraints.

The *Gripper* must not interfere with the movement of the *OT-2*. That is, the *Gripper* must not be so large or be mounted in such a way that the *OT-2*'s moving to a position that was previously reachable would cause a collision when the *Gripper* is attached.

The *Gripper* should have a large range of motion. It should be possible to collect a vial from any of the 11 deck positions. If this is not possible, the design should maximize the number of reachable deck slots.

Finally, The *Gripper* must be controllable programmatically from an *OT-2* protocol, which is written in Python.

4.1.1 *Gripper V1*

The original plan to implement the *Gripper* was to install it to the side of one of the *OT-2*'s pipettes in the location indicated by Figure 4.4. This way, the vertical movement of the pipette could be leveraged leaving only the opening and closing mechanism to be designed. Enclosing the *Gripper* in the head of the *OT-2* would also preserve its original shape and

size, leaving the machine's working area unaffected. Furthermore, with the *Gripper* in proximity to the pipette, nearly any position that is reachable by the pipette would also be reachable by the *OT-2*. Given that *Opentrons* added a margin of more than this offset, the *Gripper* could reach a vial in any slot of the *OT-2*'s deck.

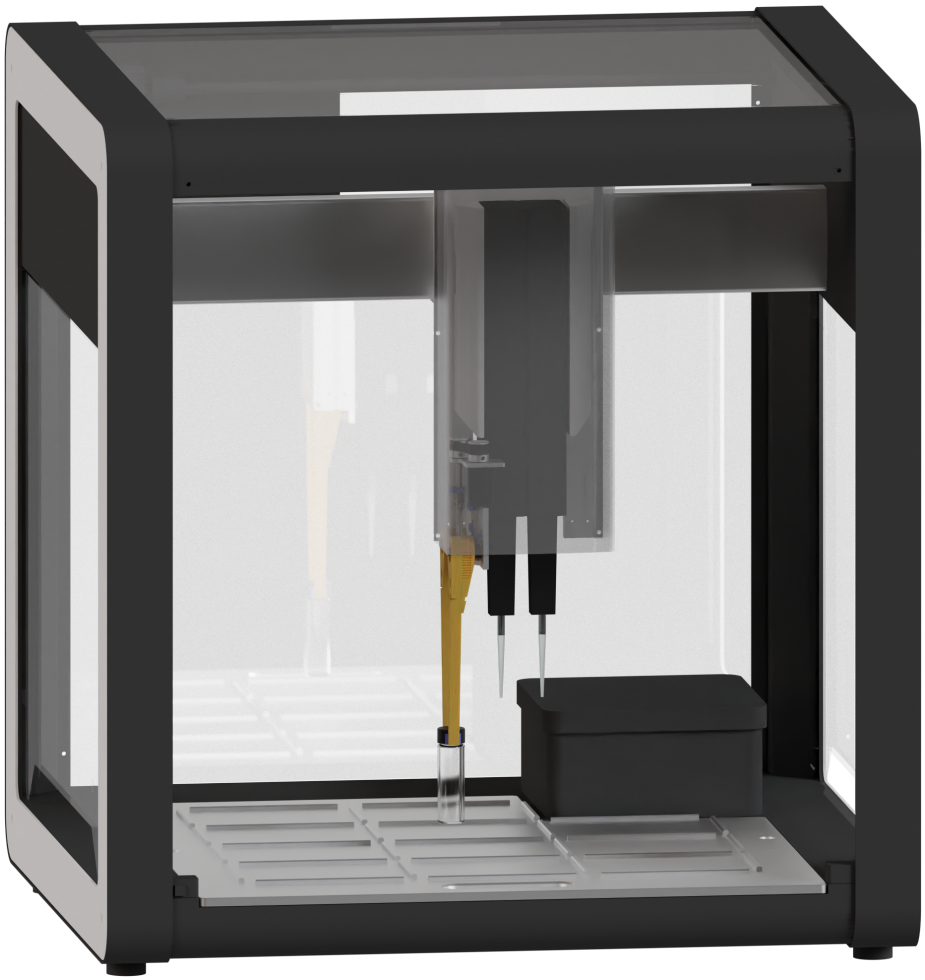


Figure 4.4: Visual aid of the intended position of the *Gripper V1* inside the *OT-2*. The pincers of the *Gripper V1* are yellow. The reference model for the *OT-2* is provided by *Opentrons* [93]. For scale, the *OT-2* is 662 mm tall. The image is created by importing the *Gripper V1* CAD model into *Blender* [95] and using this software's ray-traced rendering.

Affixing the *Gripper* to the pipette introduces certain additional constraints. The *Gripper* must be able to reach below the tip of the pipette. Otherwise, the pipette would

interfere with the actions of the *Gripper*. Figure 4.5 shows that this is satisfied.

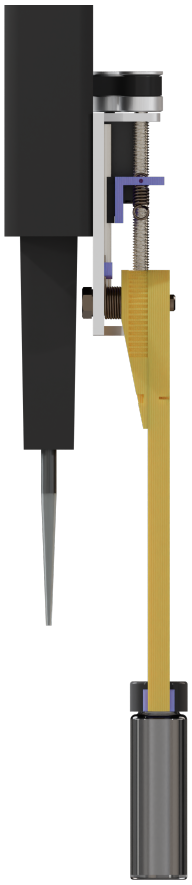


Figure 4.5: Side view of the *Gripper V1* showing that its fingers can extend below the tip of the pipette.

The *Gripper V1* features two counter-rotating fingers that are long enough to extend below the pipette's tip (constraint detailed in the previous paragraph). These are attached to an aluminium base plate via ball bearings. The fingers are actuated through a lead screw driving a sled. The sled connects to the fingers through an extension spring and section of timing belt. The purpose of the spring is to gradually add force after the finger makes contact with a container. This introduces a margin of error to the *Gripper*'s position and enables precise control of the force exerted onto the container. The lead screw is driven by a small stepper motor via a timing belt. There is a torsion spring under each finger, returning that finger to the collapsed position when the tension between the sled and fingers

is released. All of these components are shown in Figure 4.6.

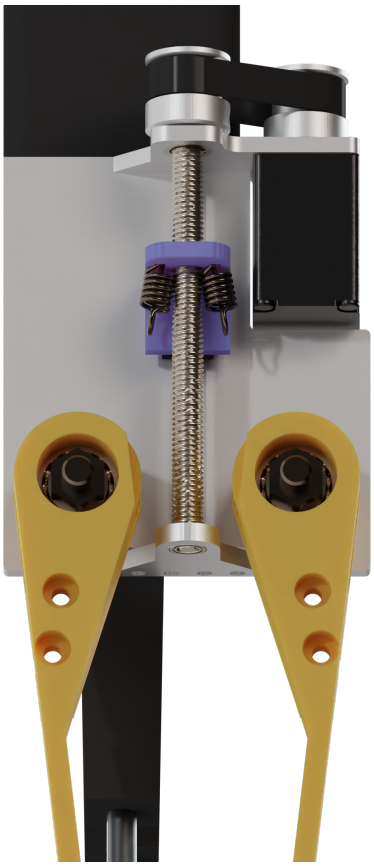


Figure 4.6: Side view of the CAD of the *Gripper V1* showing the *Gripper* fingers (yellow), lead screw (chrome), stepper motor (black), sled (purple), and timing belt (black).

The *Gripper V1* was never manufactured and has been replaced by the *Gripper V2*. The former was designed based on 3D models provided by *Opentrons* before the arrival of the *OT-2*. While the 3D model indicated that there was sufficient space in the head of the *OT-2* for this *Gripper* design, after receiving and disassembling the *OT-2*, it was discovered that this space is already taken by certain components of the robot that were missing from the reference models.

4.1.2 *Gripper V2*

The *Gripper V2* aims to solve the issues that prevent the *Gripper V1* from being completed. Namely, that the planned mounting location was already taken by components of the *OT-2*. The *Gripper V2* solves this by being mounted behind the two pipettes. The head of the robot includes T-shaped slots into which square nuts can be inserted, allowing additional modules to be bolted onto the rear of the head. The location of the *Gripper V2* in the *OT-2* is shown in Figure 4.7. Its mounting to the back of the head of the *OT-2* behind the pipettes is presented in Figure 4.8.

Unable to take advantage of the pipettes' vertical motion, the *Gripper V2* must include its own Z-axis. Thus, this *Gripper* uses a lead screw driven by a stepper motor to raise and lower itself. This was designed to maximize the vertical travel distance so that the *Gripper* can be lifted high out of the working area of the pipettes while maintaining the ability to lower sufficiently to pick a vial directly off the deck of the *OT-2*. Since no possible future module can hold a vial below the deck of the *OT-2* (without modifications to the *OT-2*), this lower bound assures that vial are reachable by the *Gripper V2* in any future custom module. The result of this is that the *Gripper V2* extends below the bottom of the head of the *OT-2* by 32.75 mm. This was unavoidable in order to achieve this large travel distance without resorting to the complexity of a telescoping mechanism.

In order to maximize the vertical travel distance of the *Gripper V2*, the total height of the stationary part of the *Gripper* (the base) is taken by linear rails. Thus, the base plate of the moving part of the *Gripper* (the head) is shaped like the letter U, allowing the carriages to extend to the limit of the rails while leaving clearance for the drive mechanism in the centre. These components are shown in Figure 4.9a.

The *Gripper* also includes a limit switch, which is used to find the zero position of the Z-axis. When the microcontroller is powered on, the position of the Z-axis cannot be known. In order to enable absolute positioning of the Z-axis, when the microcontroller starts up, the lead screw is turned until the limit switch is depressed. At this point, the Z position is set to zero. This process is typically called "homing". The limit switch is also visible in Figure 4.9a.

The movable head of the *Gripper V2* holds the two fingers and all machinery required to open and close them. Each finger is attached to a large spur gear, ensuring that they rotate symmetrically. A stepper motor is used to actuate the fingers, and a worm gear set is used to increase the torque and resolution of the motor while reducing the speed. All of these components are shown in Figure 4.9b. This axis of rotation is called the A-axis.

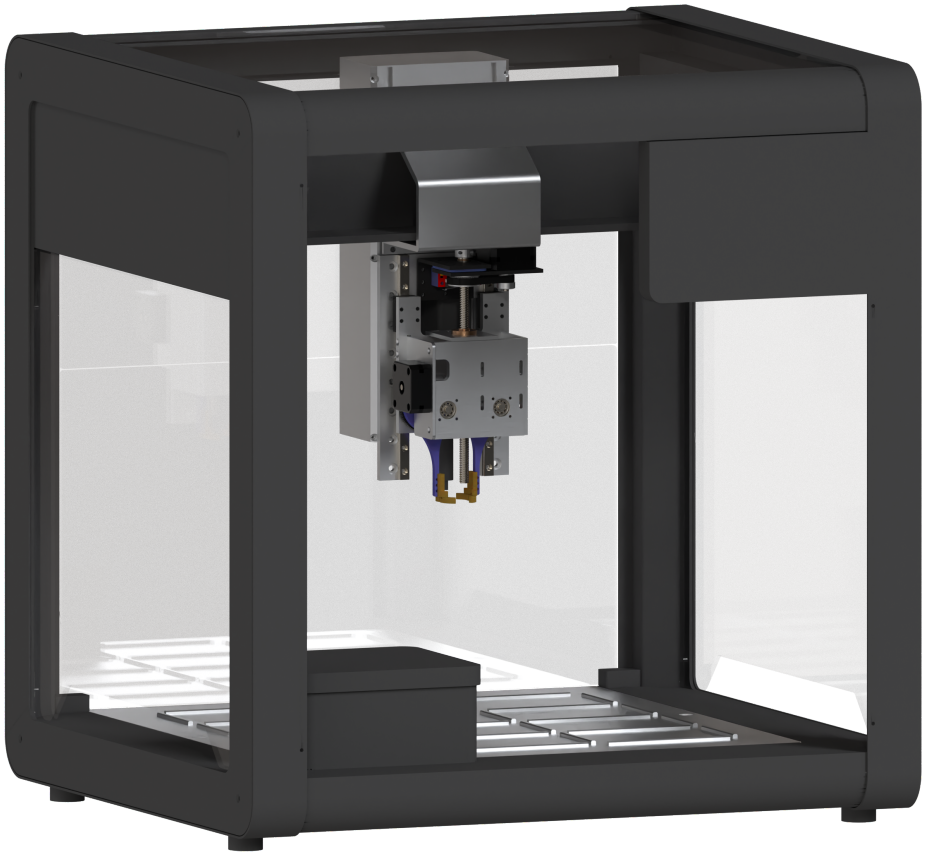


Figure 4.7: Mounting location of the *Gripper V2* inside the *OT-2* behind the pipettes attached to the head of the machine (render taken from the back of the *OT-2*). The reference model for the *OT-2* is provided by *Opentrons* [93]. For scale, the *OT-2* is 662 mm tall.

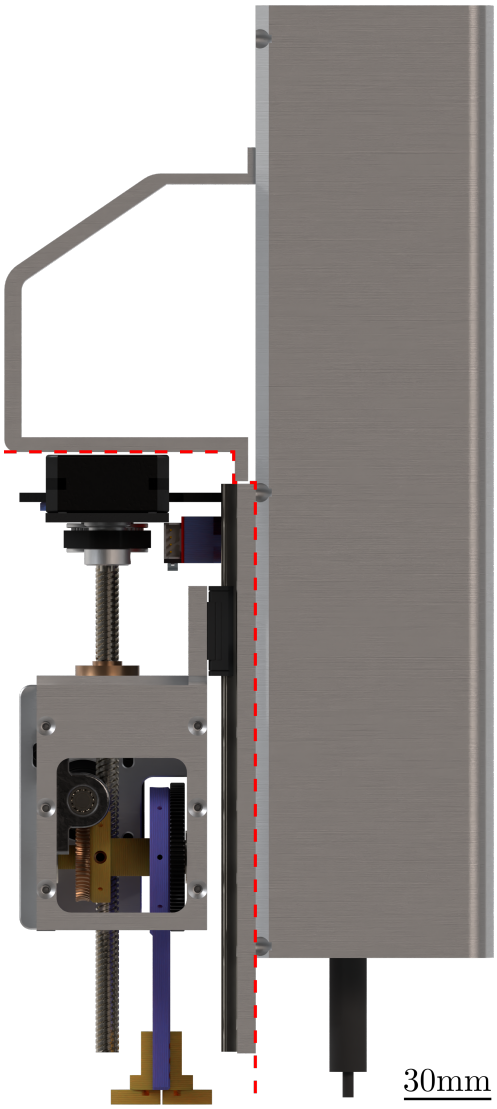
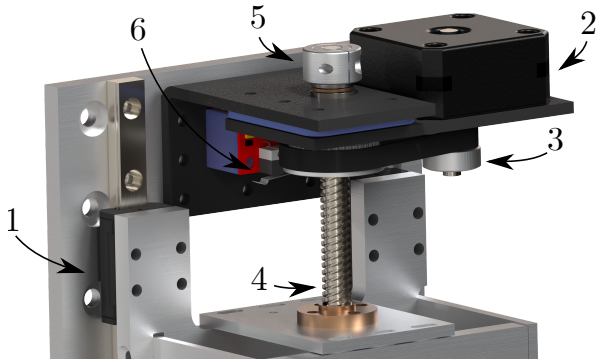
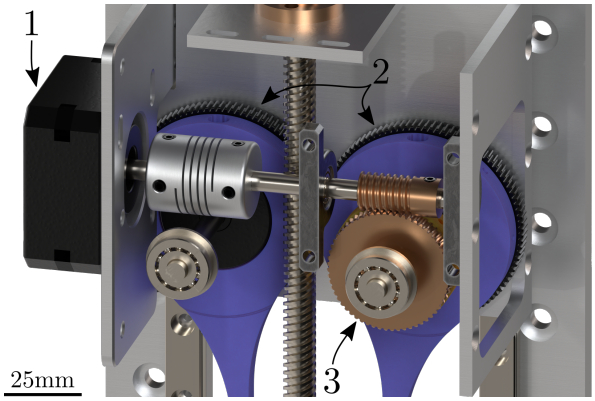


Figure 4.8: Side view of the *Gripper V2* mounting position showing that it is attached to the head of the *OT-2* behind the pipettes. All components above or right of the red dashed line are stock *OT-2* parts. The components below or left of the red dashed line constitute the *Gripper V2*.



(a) A CAD rendering of the *Gripper V2* lifting mechanism. Annotations call out the linear rails and carriages (1), the Z-axis stepper motor (2), the pulley and belt (3) connecting the stepper motor to the lead screw (4), the thrust bearing and collar supporting the lead screw (5), and the Z-axis limit switch (6).



(b) A CAD rendering of the inside of the *Gripper V2* head. Annotations call out the stepper motor to open and close the *Gripper* (1), the spur gear set (2), and the worm gear set (3).

Figure 4.9: Internal components of the *Gripper V2* with annotations indicating the most important components. The images are created by importing the *Gripper V2* CAD model into *Blender* [95] and using this software's ray-traced rendering.

4.1.2.1 Feedback for the Angle of the Fingers

The *Gripper V2*, like the *Gripper V1*, has two fingers. Each is mounted with a shaft penetrating its centre of rotation. The right finger is directly attached to its shaft; they rotate in unison. The left finger is mounted using bushings such that it can rotate independently to its shaft. The two fingers are connected via a pair of spur gears so that they open and close symmetrically.

The reason to decouple one of the fingers from its shaft is to install a torsion spring between the shaft and the fingers. This is done for the same reason as the extension springs for the *Gripper V1*: to gradually add force after the finger makes contact with a container. This introduces a margin of error to the *Gripper*'s position. Without this spring, the positioning of the *Gripper* would need to be perfect. Closing the fingers too much would exert unnecessary strain on the motor, container, or gear system. Closing the fingers too little would result in the container's slipping out of the *Gripper*. The inclusion of this spring also enables precise control of the force exerted onto the container. Rather than controlling the torque of the motor, requiring advanced electronics and a sophisticated feedback loop, it is possible to control the extension of the spring, which translates into torque.

Rotation of the left shaft translates directly to rotation of the fingers, while the right shaft represents the rotation of the motor before the spring (here, "right" and "left" are used from the perspective of the back of the *OT-2*, since the *Gripper V2* is mounted "backwards" behind the head of the machine). This is illustrated in Figure 4.10. Relative motion between the two shafts corresponds to compression or extension of the spring. By measuring the angle of each shaft, it is possible to know the angle of the fingers as well as the displacement of the spring (a proxy for torque).

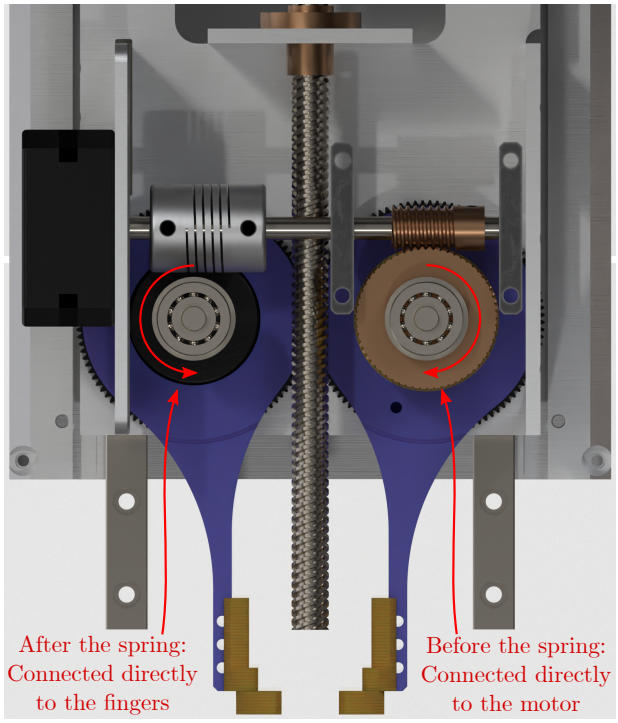


Figure 4.10: Illustration of the difference between the two shafts of the *Gripper V2*. The perspective is from the back of the *OT-2*, so the "right" finger in the image is the same as the "right" finger in the text. The image is created by importing the *Gripper V2* CAD model into *Blender* [95] and using this software's ray-traced rendering.

Thus, sensors are included in the *Gripper V2* that measure the angle of each shaft. Magnetic induction encoders are used for their ease of use, simplicity, precision, and mechanical isolation. A strong diametric magnet is attached to the end of each shaft and a custom PCB with a magnetic induction integrated circuit is mounted to the back plate of the *Gripper*. Since the two integrated circuits use the same I^2C address, an off-the-shelf I^2C multiplexer is added to allow communication to both modules. An image of the two sensor boards and I^2C multiplexer is presented in Figure 4.11.

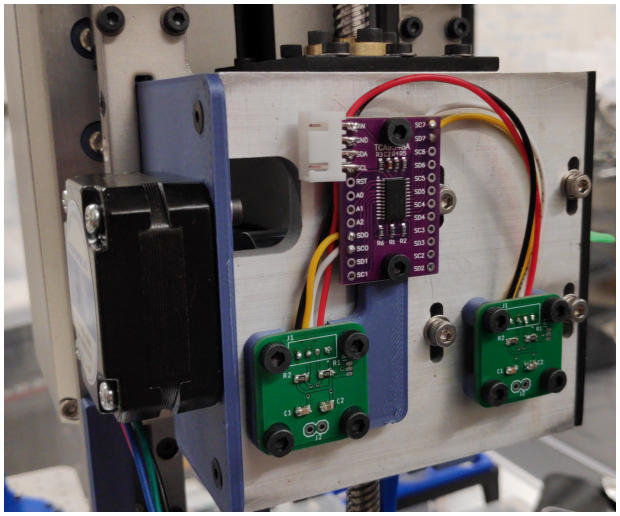


Figure 4.11: Image of the magnetic induction encoders (green boards) and I^2C multiplexer (purple board) attached to the rear of the *Gripper V2*.

4.1.2.2 Mechanical Assembly

With the CAD finished, it was possible to order the components for the *Gripper V2* and assemble this module. Many of the mechanical components were ordered from `mcmaster.com`. The custom metal panels were ordered from the University of Waterloo engineering machine shop (EMS). A photo of the fully assembled *Gripper* is presented in Figure 4.12.

4.1.2.3 Printed Circuit Board and Wiring

A printed circuit board (PCB) was designed to easily manage the wiring of all of the *Gripper*'s electronics. The heart of this board is the STM23F303K8 microcontroller. This microcontroller was chosen for this and all other custom modules for its ease of use, because it has sufficient interfaces without being overly complicated, for its cost, and because the lab is already familiar with it in the form of the NUCLEO-F303K8 development board [96].

Another notable component of the PCB is the two stepper motor drivers for the two stepper motors of the *Gripper V2*. The stepper motor driver selected is the *TMC2209* from *Trinamic* [97]. This driver was chosen for its small size, low cost, and advanced features compared to similar alternatives. Although this component is available as an integrated circuit, it was decided to include two modules following the standard 16-pin stepper driver

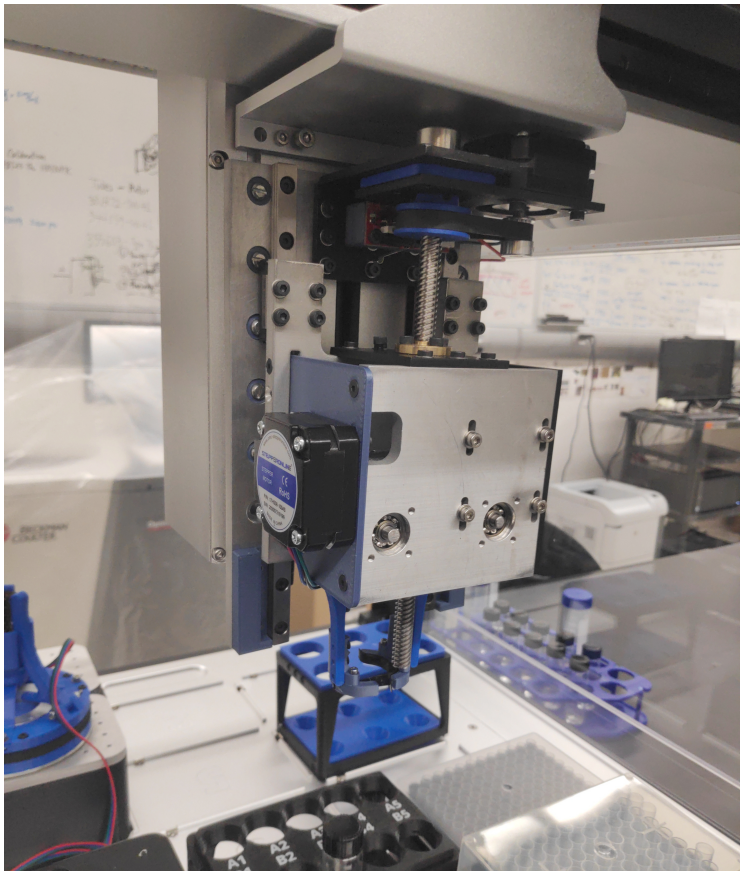


Figure 4.12: A photo of the *Gripper V2* installed behind the head of the *OT-2* after the mechanical assembly was completed.

interface that can be installed via pin headers. This alternative was selected because it is difficult to solder this integrated circuit by hand and so that a driver can easily be replaced if it is burnt out by delivering too much current to the motor.

Finally, the PCB includes a USB connector and a USB to UART converter. The USB connector provides power to the microcontroller and other low-voltage, logic-level components (there is a separate connector for the 24 V power to the stepper motor drivers). The USB connector's data pins are also connected to a *FT232RL* USB to UART integrated circuit [98]. This converter enables communication between the microcontroller and the central computer. There are also connectors for the Z-axis limit switch, JTAG programming port, I^2C bus for the magnetic induction encoders, and both motors.

This PCB was designed using *Autodesk Eagle* [99]. The schematic of this PCB is included in Appendix A and the board is included in Appendix B. An image of the soldered PCB is presented in Figure 4.13. The PCB was installed behind the head of the *OT-2* above the X-axis, which can be seen in Figure 4.14a. The reason for this location is that it is close to the *Gripper V2* electronics, because it does not move relative to the *Gripper V2*'s base, and because this does not cause a collision over the *OT-2*'s full range of motion. This location was selected before the board was designed, hence its strange shape. The location directly behind the X-axis is a more obvious position for this PCB given that it is closer to the electronics. However, there is not sufficient space when the head moves all the way to the back. The current wiring and cable-management bracket visible in Figure 4.14a already only leaves millimeters of clearance with the panel of the *OT-2*.

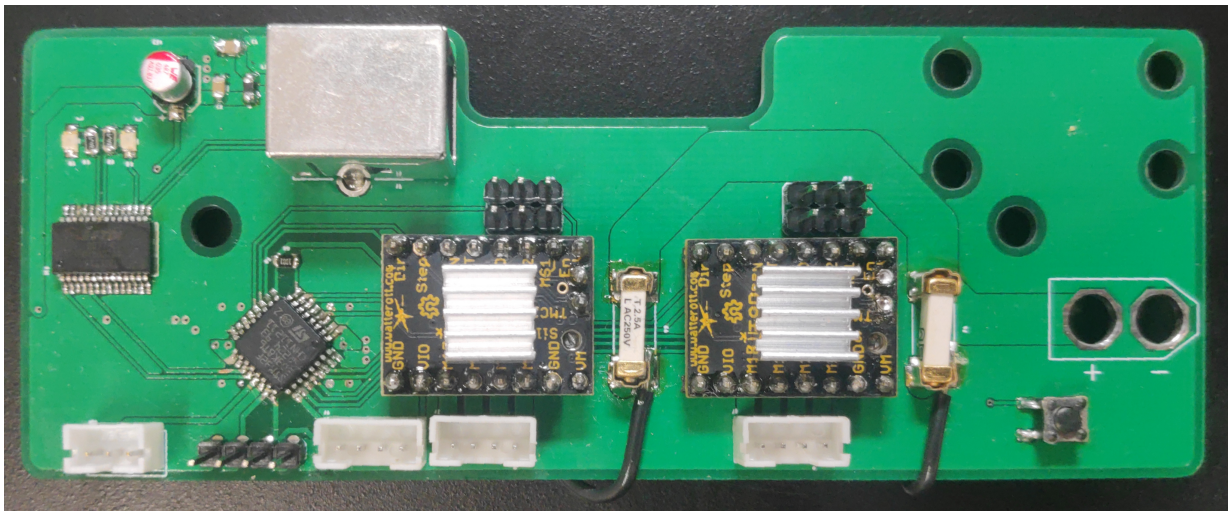
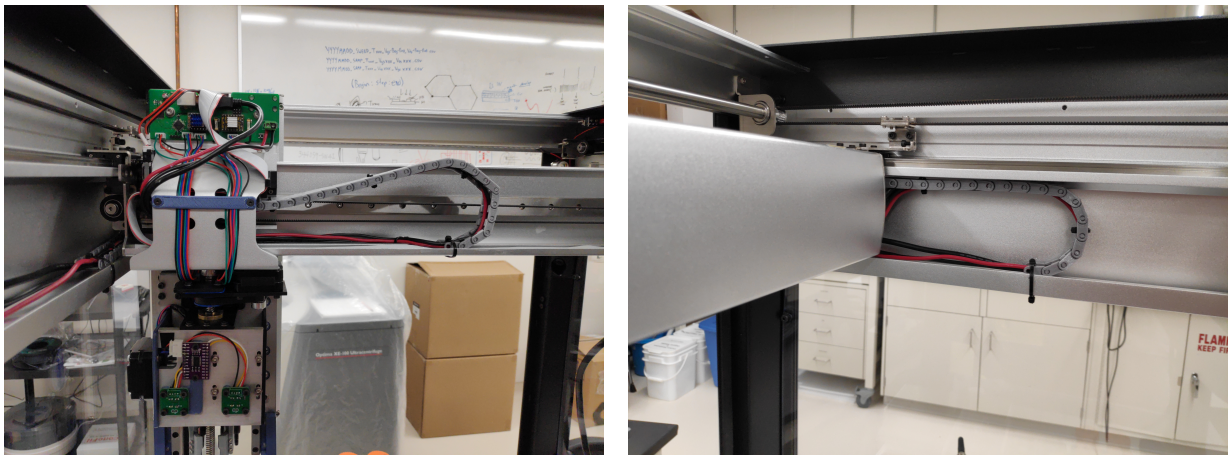


Figure 4.13: A photo of the populated PCB for the *Gripper V2*.

Four cables connect the *Gripper V2* to its PCB: two stepper motor cables, the limit switch, and the I^2C bus used by the rotary encoders. Two additional cables connect the PCB to devices external to the *OT-2*. A USB cable connects the *Gripper V2* to the *Central Command* computer. Furthermore, a pair of positive and ground wires connect this module to an external 24 V power supply. This voltage was selected because it is an industrial standard, making it easy to find components at this voltage level. A high current power supply was used so that it may power future modules as well. To avoid collisions or damage to the cables, 3D-printed cable chains are used to constrain the wires keeping them inside the cable trays of the *OT-2*. These cable chains are presented in Figure 4.14.



(a) Cable chain for the X axis of the *OT-2*

(b) Cable chain for the Y axis of the *OT-2*

Figure 4.14: Image of the 3D-printed cable chains used to route the cables required by the *Gripper V2*.

The *Gripper* also includes two additional custom PCBs as disclosed in Section 4.1.2.1. Two encoders are used to measure the rotation of the two shafts of the *Gripper V2*. These sensors use the AS5600 position sensor integrated circuit [100] and a custom PCB. These boards are extremely simple, requiring only two capacitors and two resistors in addition to the main integrated circuit. This PCB was also designed using *Autodesk Eagle* [99]. The schematic of this PCB is included in Appendix C and the board is included in Appendix D.

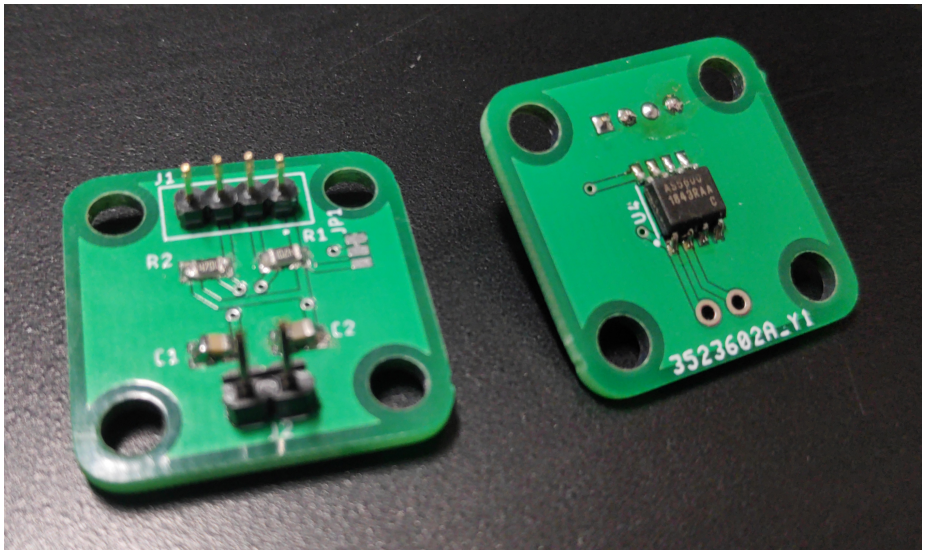


Figure 4.15: Image of two magnetic induction encoder position sensor PCBs. Both sides of the board are shown: one side includes the AS5600 integrated circuit and the other the passive components.

4.1.2.4 Software

There are two software components involved in making the *Gripper V2* work. There is a Python library for use in the *Central Command* software, and firmware running on the microcontroller of the *Gripper V2* PCB. The two components communicate with each other using UART and the USB connection discussed in Section 4.1.2.3. The messages follow a predefined command and response protocol.

The Python library is effectively a layer of abstraction to simplify the way in which commands are sent to the microcontroller. For instance, instead of manually creating the sequence of bytes that will cause the lead screw to rotate until the limit switch is pressed and sending this packet over the UART connection, it is possible to call the method `gripper.z_home()`. There are also higher-order methods — methods that call several lower-level methods — sending many commands sequentially to the microcontroller. For example, the higher-order method `gripper.pick_up(target, container_type)` picks up the container from the location specified by `target` by (1) first opening the fingers of the *Gripper*, (2) then moving the *Gripper* to the maximum height to avoid collisions while the *OT-2* is moving, (3) moving the head of the *OT-2* so that the *Gripper* is centred on the container, (4) lowering the fingers of the *Gripper* to the appropriate height for the

specific container type and for the height of the container off the deck of the *OT-2*, and finally (5) closing the fingers of the *Gripper*.

Several commands have been defined specifically for the *Gripper V2*. Each command has a code (a byte to uniquely identify the command), a list of arguments, and a response format. The communication protocol is described in detail in Section 4.6.1. The commands defined for the *Gripper* are enumerated in Table 4.1. The *Gripper* also supports all base commands defined in Table 4.13.

Table 4.1: Command names, codes, and functions specific to the *Gripper V2*.

Command Name	**Code**	**Function**
`SET_STEPPER_ENABLE`	`0x16`	Enable or disable either of the *Gripper*'s stepper motors.
`READ_SPEED_ACCEL`	`0x01`	Read the current speed and acceleration setting of one of the *Gripper*'s stepper motors.
`WRITE_SPEED_ACCEL`	`0x19`	Update the speed and acceleration setting of one of the *Gripper*'s stepper motors.
`Z_MOVE_TO`	`0x10`	Move to a specific position in the Z-axis (a specific height).
`Z_CHECK_POSITION`	`0x11`	Read the current Z-axis position.
`Z_CHECK_ENDSTOP`	`0x12`	Check whether the Z-axis limit switch is depressed.
`Z_HOME`	`0x13`	"Home" the Z-axis by rotating the lead screw until the limit switch is depressed.
`Z_READ_LIMITS`	`0x17`	Read the soft limits of the Z-axis (Section 4.6.1.2 §3).
`Z_WRITE_LIMITS`	`0x18`	Update the soft limits of the Z-axis (Section 4.6.1.2 §3).
`A_READ_ANGLES`	`0x14`	Read the current position of both magnetic induction rotary encoders.
`A_APPLY_FORCE`	`0x15`	Apply a force between the fingers by rotating the motor until the desired spring extension (difference between two angle sensors) is achieved.

Like all modules, the *Gripper* firmware keeps track of a status variable. This is used to communicate the state of the *Gripper* and any errors encountered. The commands relating to the status are defined in Section 4.6.1. The *Gripper* status is implemented as an enumeration type [101] whose variants are defined in Table 4.2.

Table 4.2: Variants of the *Gripper V2* status enumeration.

Variant	Value	Description
`OK`	0	There is no error and the microcontroller is ready to receive new commands.
`WORKING`	1	The previous command issued is still in progress.
`OVERTRAVEL`	2	The A-axis motor has moved beyond the safe limit. An error status is entered to avoid damage.
`UNDERTRAVEL`	3	Unused.
`RUNAWAY`	4	The stepper motor is commanded to move but there is no change in the rotary encoder, indicating a possible motor wiring problem or mechanical disconnection.
`STEPPER_ERROR`	5	An error is reported by the stepper motor library (Section 4.6.1.2 §6).
`OVERFORCE`	6	The force (difference between two rotary encoders) is above the safe limit.

The *Gripper* firmware uses the Mbed [102] hardware abstraction layer and is written in C++. The firmware uses the communications library (Section 4.6.1.1), the asynchronous stepper library (Section 4.6.1.2), and the rotary encoder library (Section 4.6.1.3).

4.2 *Vial Opener*

The purpose of the *Vial Opener* is to automatically remove the cap of vials and tubes so that liquid can be added or extracted, and to re-apply the cap, sealing the liquid inside. It is important to seal vials when liquid is not presently being transferred into or out of them to reduce the chances that contaminants enter the vial or that the liquid is spilled accidentally by the automated system.

4.2.1 Mechanical Design and Assembly

The *Vial Opener* must be able to remove and re-apply the cap of a vial or tube. Since the *Gripper* will be bringing the container to and from the *Vial Opener*, an easy way to remove the cap is to hold it in the *Gripper* while the base of the container is rotated in the *Vial Opener*. Rotating the container in the *Vial Opener* clockwise while the lid is kept still will

unscrew the threaded cap, removing it. Symmetrically, rotating the cap counter-clockwise will tighten the threads and re-apply the cap. Therefore, the *Vial Opener* must be able to latch onto the container, in which case rotation of the *Vial Opener* is transmitted into rotation of the container. Additionally, it must be able to release the container, letting it go free. The computer model of the module is presented in Figure 4.16.

A major complication of the *Vial Opener* is that whatever mechanism is used to grasp the vial must also be rotated. If electronic actuators are used, the wires would become tangled when the vial is rotated. This problem can be solved in many ways: a slip ring could be used to transfer electrical connections from the stationary side to the rotating side, or a mechanical system could transfer the grasping and releasing motion from an actuator in the stationary part. The latter was chosen for simplicity and for reliability. A solenoid (an electromagnetic linear actuator) is used to transmit a force through the centre of the large bearing used for the vial's rotation. This solenoid pushes on the three pincers used to grasp the vial (Figure 4.17a). The force is applied far from where the pincers are hinged, creating a lever arm that rotates them. It was noticed that the *Vial Opener* is always locked while rotating, so the releasing mechanism is only activated when the vial is stationary, which simplifies matters. Thus, the fingers default to the locked position so that no continual force is required during rotation. When the vial comes to a stop and must be released, the solenoid is activated to push the fingers away from the vial, releasing their grip. A visualisation of this is provided in Figure 4.18.

The rotating part (rotor) of the *Vial Opener* is built around a large bearing. The centre of the bearing is attached to the solenoid and the stationary section of the *Vial Opener*, while the outside engages with the rotor. The latter supports the cup designed to hold the vial, serves as the hinge point for the three fingers, and has teeth around its perimeter to engage with a timing belt. A thin disk is bolted to the rotor after the bearing has been inserted, securing this connection. All of these pieces are called out in Figure 4.17a. After the rotor is assembled, the centre of bearing is affixed to the solenoid. A piece in the shape of an upside-down top hat is threaded onto the solenoid, compressing the bearing against the shoulder of the solenoid, locking it into place. Compressed between the solenoid shoulder and bearing is the retaining disc, which is first bolted to the solenoid plate (Figure 4.17c), securing the solenoid assembly and rotor assembly. All of these pieces of the solenoid assembly are presented in Figure 4.17b. The aforementioned timing belt is connected to the ubiquitous NEMA 17 [103]–[105] stepper motor that spins the rotor. A stepper motor is used for its easy speed control and this specific model is selected for its availability, affordability, and size. The base assembly includes a large 3D-printed piece that interlocks with two slots of the *OT-2* deck. Onto this piece two machined aluminium plates are bolted. The first is the aforementioned solenoid plate. The second affixes the

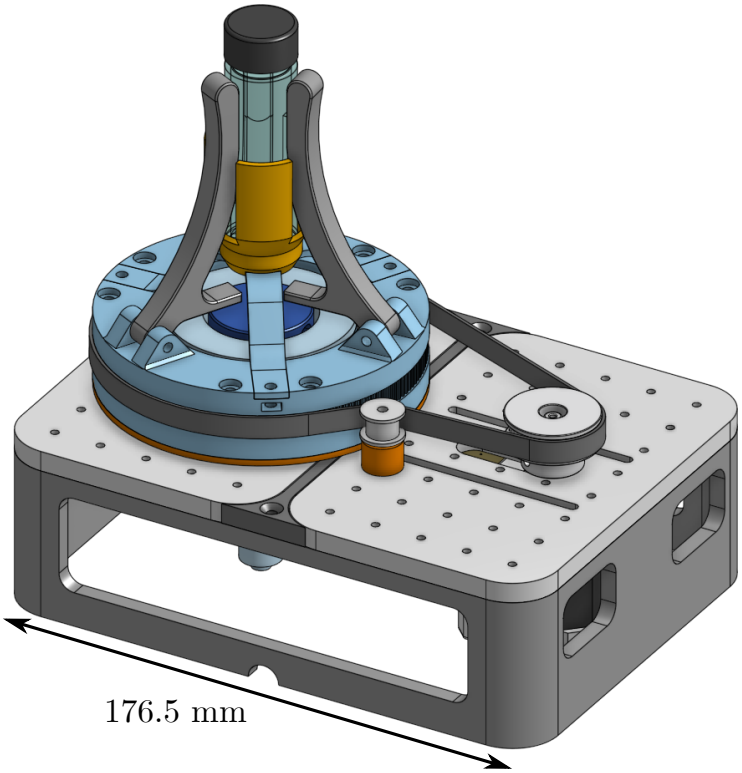
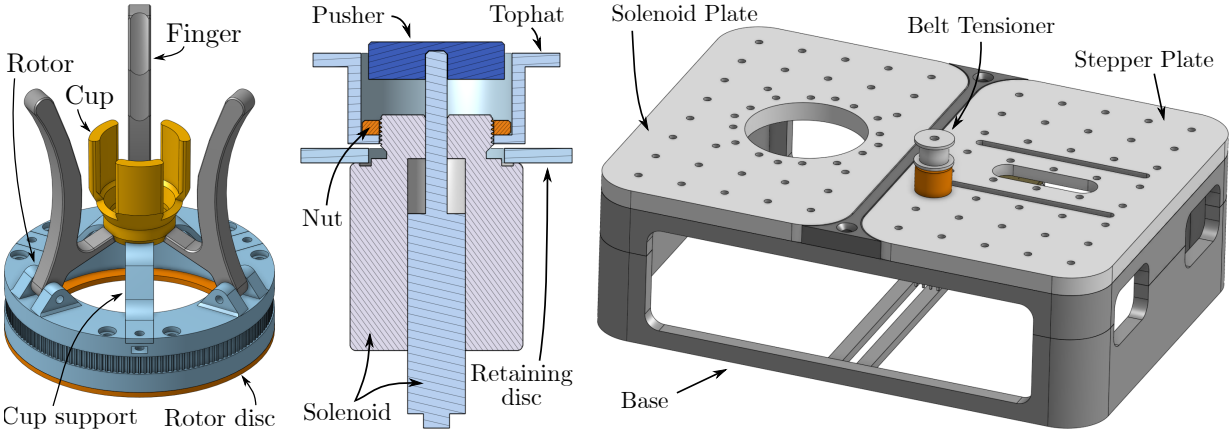


Figure 4.16: CAD screenshot of the *Vial Opener*.

stepper motor through two long slots, allowing the position of the motor to be adjusted in order to properly tension the timing belt. The two machined plates include a grid of threaded holes for any possible future expansion. Indeed, this grid was invaluable to add an idler pulley when the belt was too loose. All of these pieces of the base assembly are presented in Figure 4.17c.

The "cup" of the *Vial Opener*, presented in Figure 4.17a, is a small 3D-printed part that holds the vial or tube between the pincers in the centre of rotation. It is designed to support both the 14.7 mL glass vial and the 50 mL polypropylene tube. Therefore, it has a flat bottom to support the vial out of which a cone is cut to support the tube. There are also walls to hold up the taller and pointy-bottomed tube, which are not needed for the short and stable vial. The cup is supported by three thin 3D-printed S-shaped arms connected to the rotor called the "cup supports". Figure 4.19 shows how this cup can be used to hold a vial and a tube.

The five pieces to be machined — the solenoid plate, the stepper plate, the top hat, the



(a) Rotor assembly (b) Solenoid assembly (c) Base assembly

Figure 4.17: Annotated index of the pieces of each sub-assembly of the *Vial Opener* module. The sub-figures are not to scale.

rotor disc, and the solenoid retaining disc — were all milled and turned at the University of Waterloo Engineering Student Machine Shop. Once the metal parts were machined, the plastic parts 3D-printed, and the solenoid, motor, and fasteners delivered, the *Vial Opener* was assembled. The result is presented in Figure 4.20.

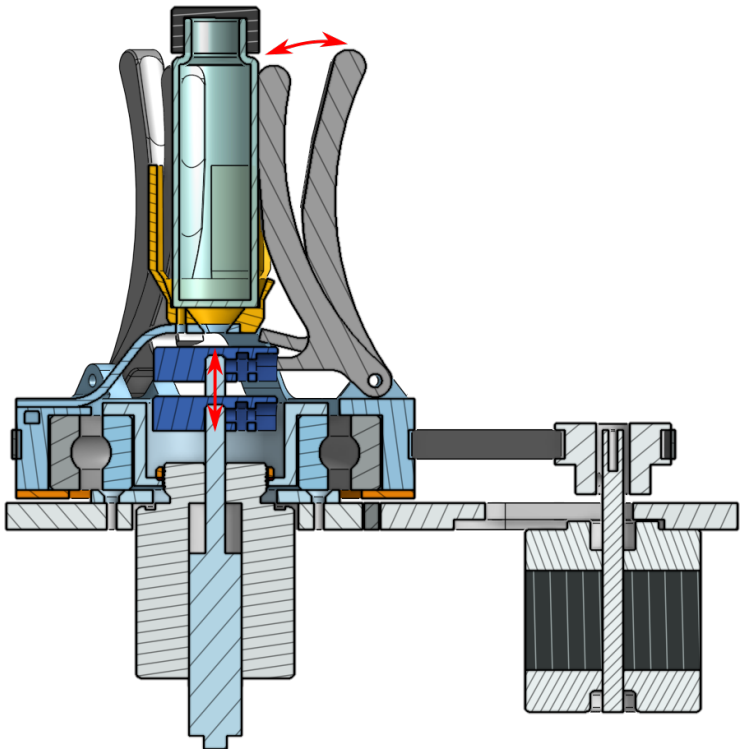
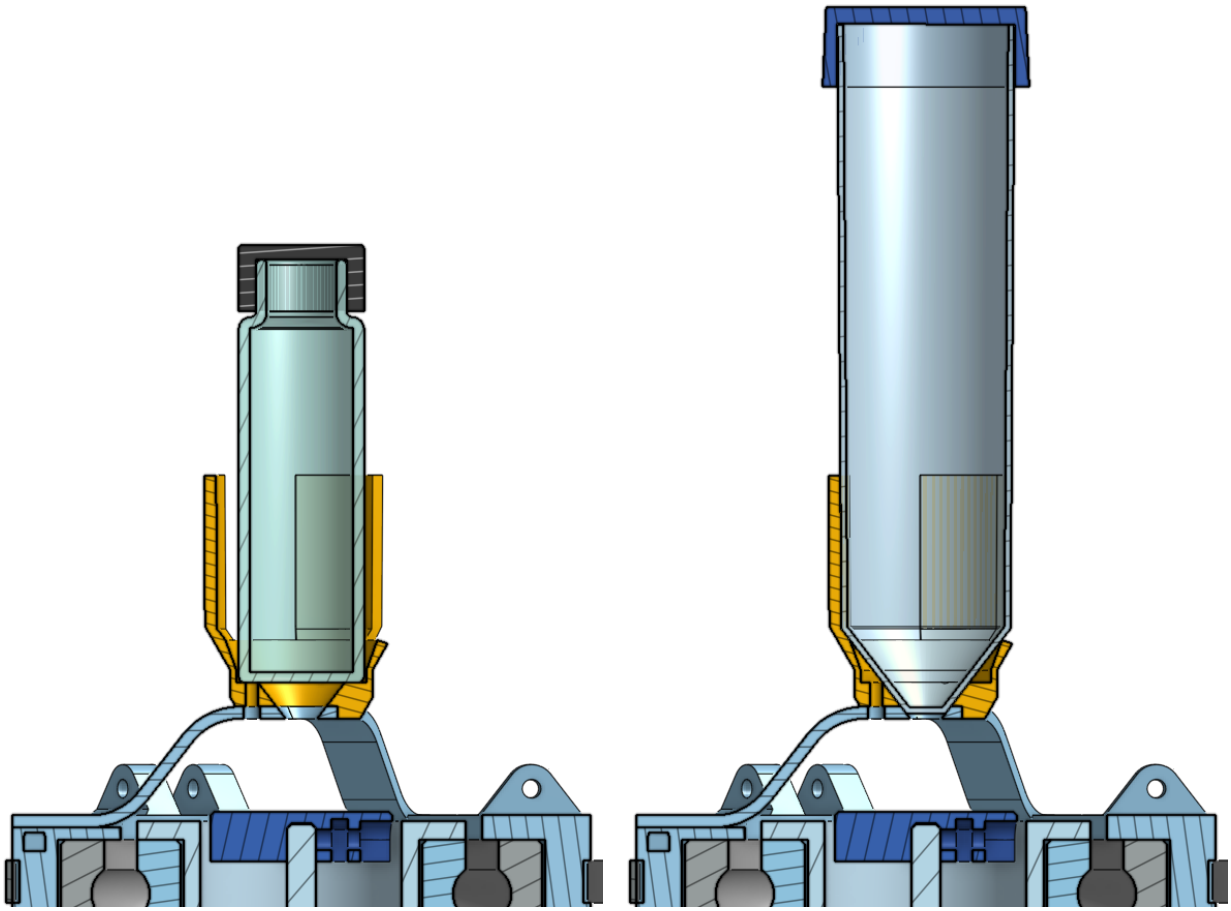


Figure 4.18: Cross sectional view of the mechanism to grasp and release the vial. The mechanism is shown in both positions simultaneously to illustrate the intended movements. The dark blue attachment on the end of the solenoid pushes the dark gray fingers away from the vial.



(a) Cup holding a vial.

(b) Cup holding a tube.

Figure 4.19: Cross sectional view showing the cup of the *Vial Opener* (orange) in each supported configuration.

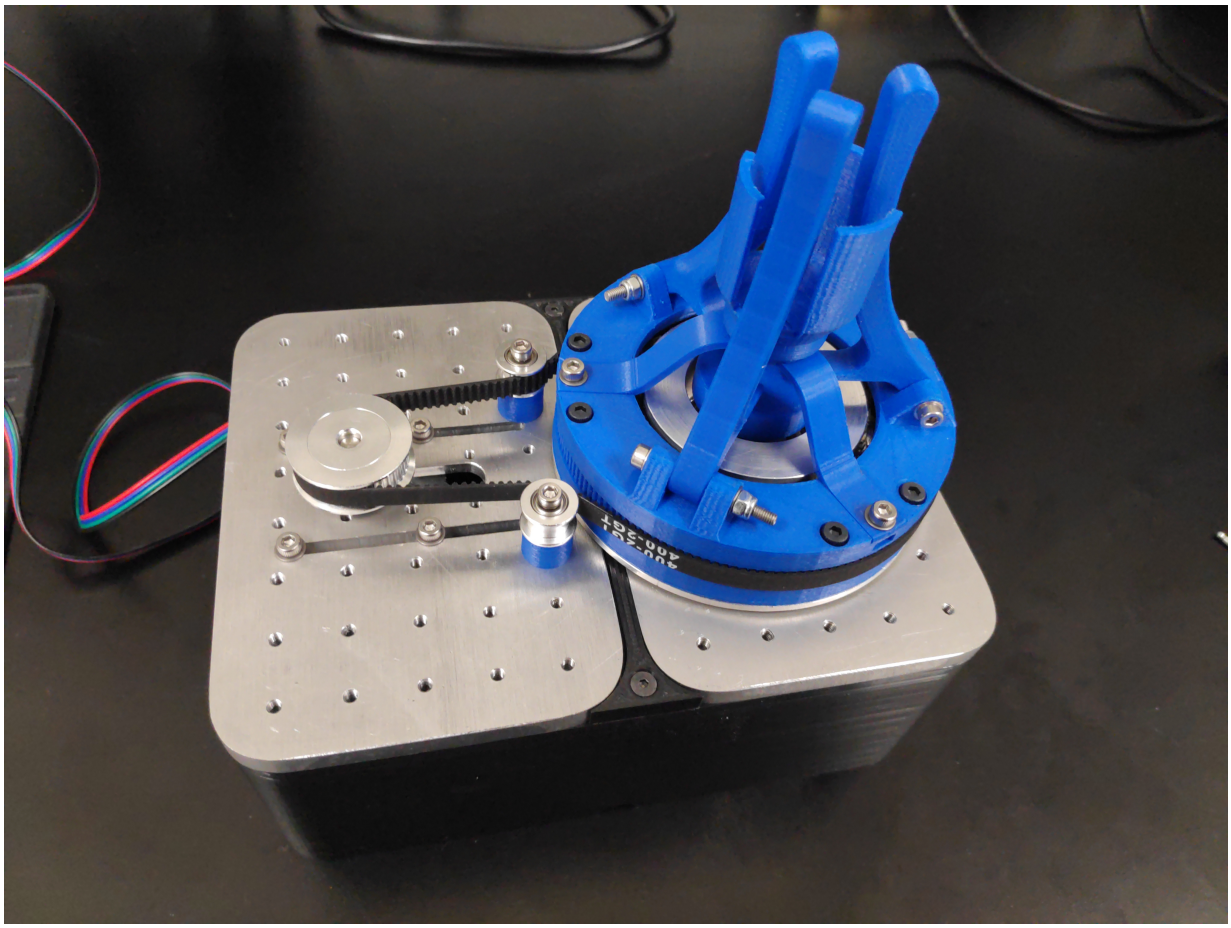


Figure 4.20: A photo of the fully-assembled *Vial Opener*.

After the initial assembly of the module, there was a need to add a cap storage system. After the *OT-2* was delivered, it was discovered that the brim of the vial inside the *Vial Opener* is above the maximum height of the liquid handling robot's pipettes. Thus, it is not possible to transfer liquids into the vial while it is in the *Vial Opener*, and it is not possible to transport the vial back to its rack when the gripper is occupied by the cap. Transporting the vial back to its rack also improves efficiency since the source of liquid transfers into the vial is either another vial in the same rack or a stock from a tube rack in an adjacent slot of the *OT-2* deck (Figure 4.3). This cap storage system was easily added with 3 simple 3D-printed pieces that are bolted to the exiting grid of tapped holes in the metal plates of the *Vial Opener*. This addition, which is presented in Figure 4.21, enables the caps of 12 vials to be stored simultaneously.

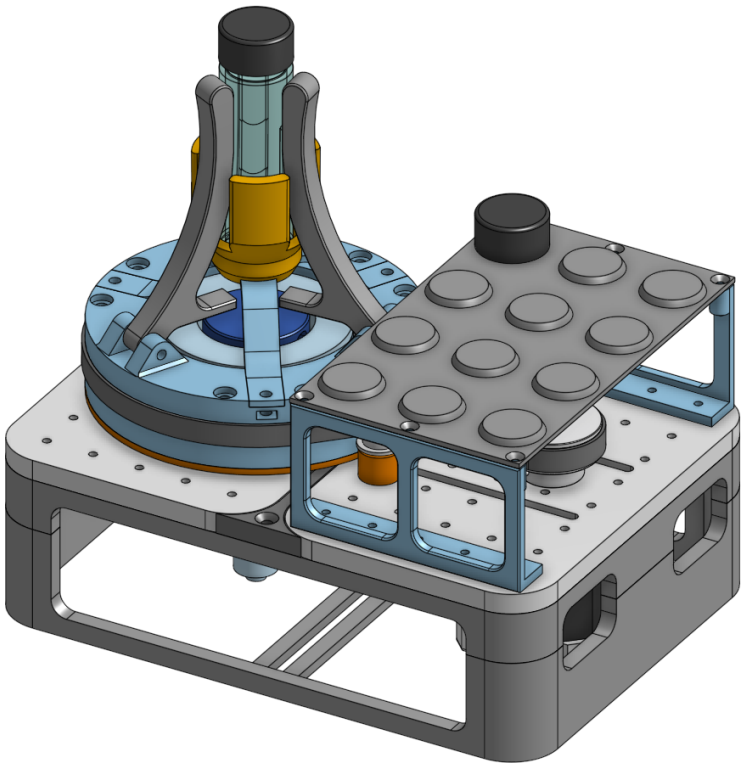


Figure 4.21: CAD screenshot of the *Vial Opener* with cap storage addition.

4.2.2 Printed Circuit Board and Wiring

A printed circuit board (PCB) is included inside the *Vial Opener* in order to simplify the wiring and to provide all the required electronics in a compact package. The primary functions of this board are to control the stepper motor responsible for the rotation of the *Vial Opener* and the solenoid responsible for releasing the vial. Like the other custom modules, the heart of this board is the STM23F303K8 microcontroller. This microcontroller was chosen for its ease of use, because it has sufficient interfaces without being overly complicated, for its cost, and because the lab is already familiar with it in the form of the NUCLEO-F303K8 development board [96].

The stepper motor driver selected is the *TMC2209* from *Trinamic* [97]. This is the same as the *Gripper* and all other custom modules using stepper motors. The driver is included as a daughter board rather than as an integrated circuit soldered directly onto the *Vial Opener* PCB. This removes the need to manually solder such a small component and allows for the driver to easily be replaced as needed.

The solenoid is activated by connecting its two wires to the 24 V power supply. This connection can be enabled and disabled programmatically from the microcontroller through the use of a solid state relay. In fact, there are two solid state relays and two 24 V digital outputs for redundancy.

Finally, the *Vial Opener* PCB includes a USB connector and a USB to UART converter. This connection is used to power the microcontroller while the USB data pins are connected to a *FT232RL* USB to UART integrated circuit [98]. This converter enables communication between the microcontroller and the central computer.

This PCB was designed using *Autodesk Eagle* [99]. The schematic of this PCB is included in Appendix E and the board is included in Appendix F. An image of the soldered PCB is presented in Figure 4.22. The PCB was installed inside the *Vial Opener* between the solenoid and the stepper motor.

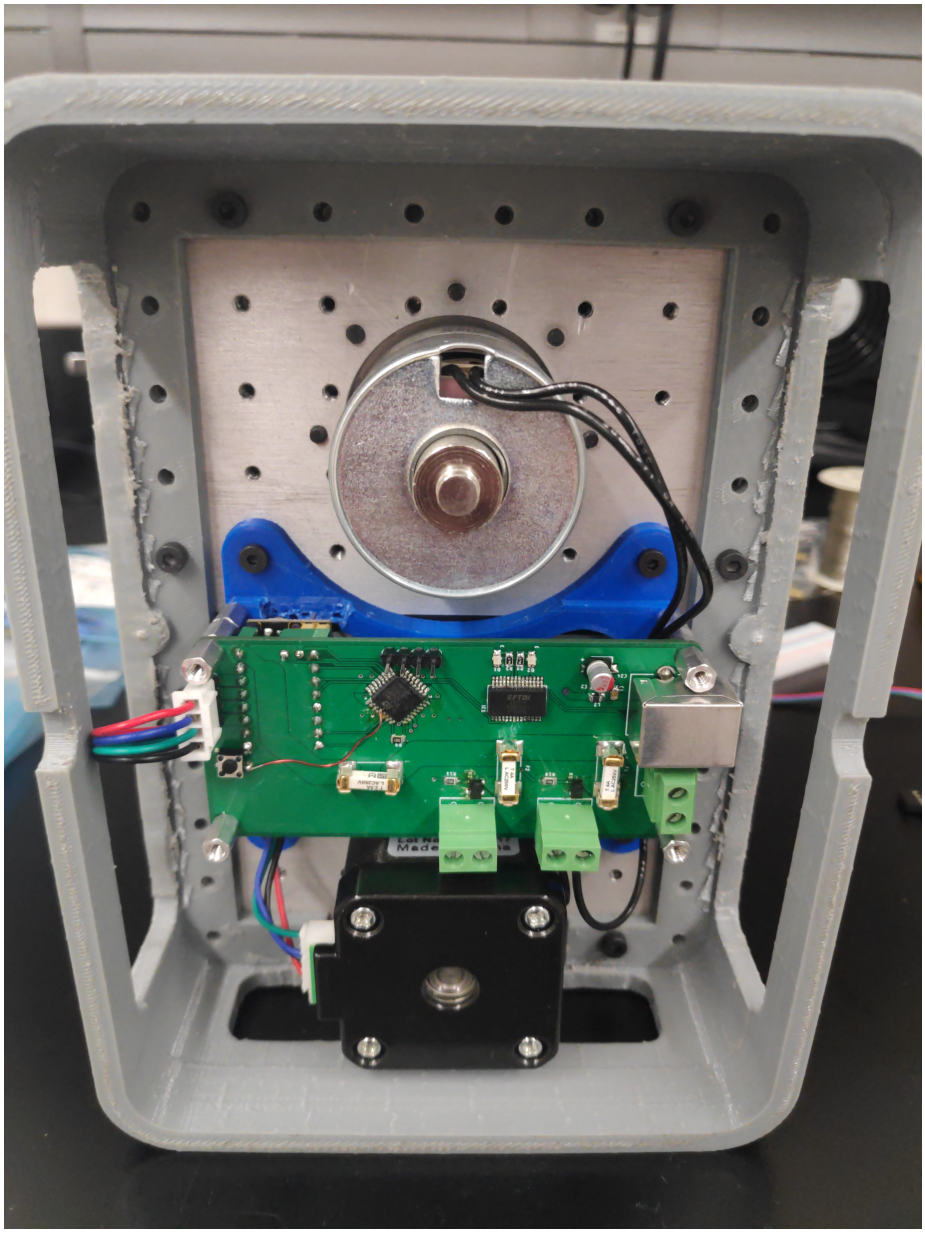


Figure 4.22: Image of the PCB wired into and installed inside the *Vial Opener*.

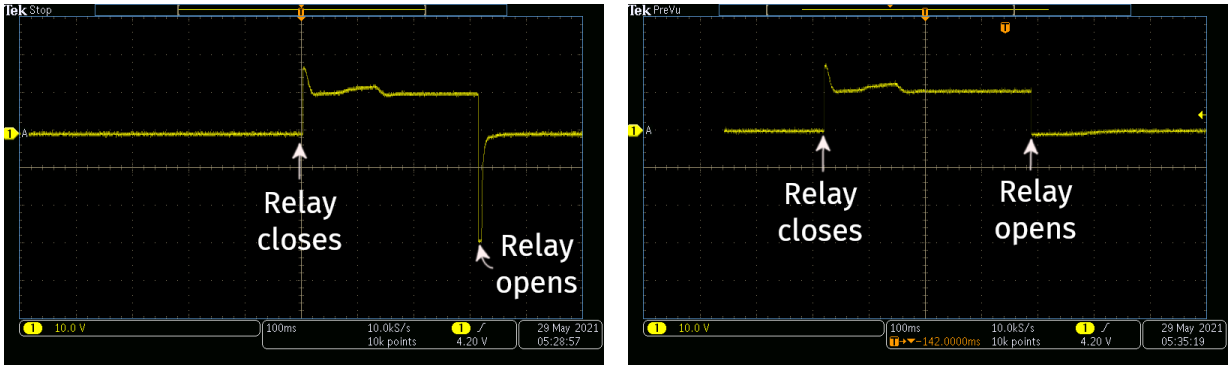
4.2.2.1 Mistakes

Several mistakes were made in the design of this PCB. First, the reset and boot selection pins of the microcontroller were left disconnected. However, these two play an important role when programming the microcontroller. As a consequence, a button was delicately connected to the reset pin via a small piece of thin, enabled wire. This button is soldered to a ground pin of the stepper motor driver. This is not an elegant solution, but is easily fixed in a new revision of the board. Furthermore, this is a lesson in not leaving pins unconnected. If they are not used in the design, it is best to route them to a pin header just in case they are useful for a future addition.

Second, the stepper motor pins were mirrored in the design. As a result, the stepper motor driver is on the back of the PCB, requiring long standoffs since a large clearance is needed on both sides. In fact, it is possible to find devices that use both standards, so this implementation is not "wrong". It is possible to solder the pins on the other side of the daughter board to accommodate this mirrored socket. However, it was decided that it is better to have a single standard between all PCBs rather than to have the driver on the front of the *Vial Opener* PCB. This is another mistake that is easily fixed by a second revision.

Third, flyback diodes were omitted from the design. Such diodes are required to dissipate the energy that builds up in the solenoid, which is essentially a large inductor, when it is disconnected. Without a flyback diode, when the solenoid goes from powered to disconnected, a large negative voltage will appear between the device's two wires, which can cause damage to the circuit board components. This is illustrated in Figure 4.23, which shows oscilloscope traces with and without the flyback diode. However, this is easily fixed by soldering a diode underneath the screw terminal between the two contacts. This diode provides a path for the current to dissipate when the solenoid is disconnected. Again, a flyback diode can easily be added in a second revision.

All of these changes were implemented and are included in the schematic and board of Appendix E and Appendix F, respectively.



(a) Without a flyback diode, there is a large negative voltage spike of around -30 V when the solenoid is disconnected.

(b) With the flyback diode, the negative voltage spike is eliminated.

Figure 4.23: Oscilloscope traces illustrating the need for a flyback diode in the circuit disconnecting the solenoid.

4.2.3 Software

The *Vial Opener* has two software components like all custom modules: there is the microcontroller firmware written in C++ and using the Mbed framework [102] and the Python library acting as a layer of abstraction making the module easier to use. The two components communicate with each other using UART and the USB connection discussed in 4.2.2. The messages follow a predefined command and response protocol (Section 4.6.1.1).

The Python library is used to facilitate the communication with the firmware. This layer of abstraction allows one to use the method `vial_opener.move_to()` to set the position of the rotor rather than manually creating the sequence of bytes with the same effect and sending this packet over the UART connection.

Several commands have been defined specifically for the *Vial Opener*. Each command has a code (a byte to uniquely identify the command), a list of arguments, and a response format. The communication protocol is described in detail in Section 4.6.1. The commands defined for the *Gripper* are enumerated in Table 4.3. The *Vial Opener* also supports all base commands defined in Table 4.13.

Table 4.3: Command names, codes, and functions specific to the *Vial Opener*.

Command Name	Code	Function
`SET_STEPPER_ENABLE`	`0x16`	Enable or disable the *Vial Opener*'s stepper motor.
`WRITE_SPEED_ACCEL`	`0x19`	Update the speed and acceleration setting of the *Vial Opener*'s stepper motor.
`A_MOVE_TO`	`0x10`	Move to a specific position in the A-axis (rotary axis).
`A_CHECK_POSITION`	`0x11`	Read the current position in the A-axis (rotary axis).
`READ_RELAY`	`0x12`	Read the current state (enabled/disabled) of one of the *Vial Opener*'s solid state relays.
`WRITE_RELAY`	`0x13`	Enabled or disabled of one of the *Vial Opener*'s solid state relays.

Since the *Vial Opener* has long-running commands (`A_MOVE_TO`), it must keep track of a status variable so that these commands can run asynchronously (see Section 4.6.1.1 and Section 4.6.1.2). The commands relating to the status are defined in Section 4.6.1. The *Vial Opener* status is implemented as an enumeration type [101] whose variants are defined in Table 4.4. This module does not have elaborate error conditions, so only implements the base statuses and the stepper motor library statuses.

Table 4.4: Variants of the *Vial Opener* status enumeration.

Variant	Value	Description
`OK`	0	There is no error and the microcontroller is ready to receive new commands.
`WORKING`	1	The previous command issued is still in progress.
`STEPPER_ERROR`	5	An error is reported by the stepper motor library (Section 4.6.1.2 §6).

4.2.4 Coordination with the *Gripper*

The proper operation of the *Vial Opener* depends on the *Gripper*. The *Gripper* is required to prevent the cap of a container from rotating while the *Vial Opener* rotates the base of the container, fastening or loosening the threads between the cap and base. In fact, the distance between the cap and container changes as the threads are turned. Tightening

the cap brings it closer to the container, while loosening the cap pushes it away from the container. Therefore, the *Gripper* is required to prevent the cap from rotating, but must also move the cap vertically in careful coordination with the *Vial Opener*.

This is possible with the *Gripper* and *Vial Opener* firmware as described, but requires certain considerations in the Python library. First, the movement of the two modules must be well synchronized. The motion must start at the same moment and the speed and acceleration must be coordinated. This is accomplished by calling `gripper.z_move_to` and `vial_opener.a_move_to` in two different threads to enable parallel processing.

Second, the amount of movement and relative speed must be precisely calculated based on the pitch of the threads for that specific container. Only two containers are supported in the current ATPE system, the 50 mL tube and the 14.7 mL vial, which have a thread pitch of 6 mm and 3.3 mm, respectively. This was determined by careful measurement and experimentation with this coordination effort. These numbers are used to calculate the travel distances, speed, and acceleration of the *Gripper* based on those of the *Vial Opener*.

The coordination is implemented separately from the *Vial Opener* library, since it also needs access to the *Gripper*. This new module has only two public functions: `remove_cap` and `replace_cap`. These both execute an underlying generalized command with different arguments for each case. This function calculates the travel distance, speed, and acceleration of the *Vial Opener* and the *Gripper*. Two target positions are saved for the *Gripper*: one with the cap firmly on the container and the other with the cap above the container. These positions are identical for the `remove_cap` and `replace_cap` functions except that the order is inverted in the second case. Then, the *Gripper* is moved to the zero position and the *OT-2* is moved so that the *Gripper* is centred on the *Vial Opener*'s axis of rotation. The *Gripper* is lowered to the first saved position and, in the case of `remove_cap`, closes its fingers. Finally, the coordinated motion begins. The *Vial Opener* is commanded to rotate a fixed number of turns while the *Gripper* is commanded to move to the second saved position. These movements begin at precisely the same moment with the relative maximal speed and acceleration directly proportional to the thread pitch.

Following the addition of the aforementioned cap storage system, two more public functions were added to this coordination Python module: `remove_and_store_cap` and `replace_cap_from_storage`. These call `remove_cap` and `replace_cap`, respectively, but use the *Gripper* to deposit the removed cap in the storage system in the former case and pick up the appropriate cap from the storage system before replacing it in the latter case. The position of the caps is tracked by the sample's barcode, ensuring that the correct cap is replaced on the vial. The occupied storage spaces are tracked in this way as well, so `remove_and_store_cap` will automatically select the next empty spot.

This coordination has been tested in isolation and works well. However, unfortunately, the *Vial Opener* has not yet been used successfully in ATPE protocols. The reason for this is that the cap storage system was only added after the hybrid ATPE attempts. However, the system has been tested in isolation and nothing prevents its use in the next attempt of hybrid ATPE.

4.3 Separation of the Two Phases

The automatic separation of the top and bottom phases is vital for unattended ATPE. A prerequisite of this is detecting the boundary between the two phases so that the *OT-2* can be used to extract the top phase. Because of the variable height of the boundary and the numerous possible colours of each phase, the problem of detecting the boundary is challenging. This is solved using computer vision. A photo of the vial containing two phases is captured and image processing software attempts to locate the boundary. This assumes that the location of the boundary in the image can be mapped to some height from the bottom of the vial. This is a reasonable assumption. Even if the relationship is not linear, which could be the case due to perspective, camera distortions, or a misalignment, it would be possible to create a lookup table to map between the two.

In order to take reliable photos with consistent conditions, the *Photo Stage* is implemented. This box holds the vial in a repeatable position and provides consistent lighting. The *Photo Stage* is detailed in subsection 4.3.1.

4.3.1 *Photo Stage*

The purpose of the *Photo Stage* is to provide an environment where photos of vials can be taken with consistent conditions to be processed by computer vision in order to located the phase boundary. The *Photo Stage* is essentially a box that blocks out external light while providing repeatable lighting through the use of internal LEDs. There is a stand for a vial and a motorized lid that allows the *Gripper* of the *OT-2* to deposit a vial into this stand. Finally, there is a camera to take photos of the vial. An image showing the internal components of the *Photo Stage* is given in Figure 4.24. A photo taken inside the *Photo Stage* using its camera is presented in Figure 4.25.

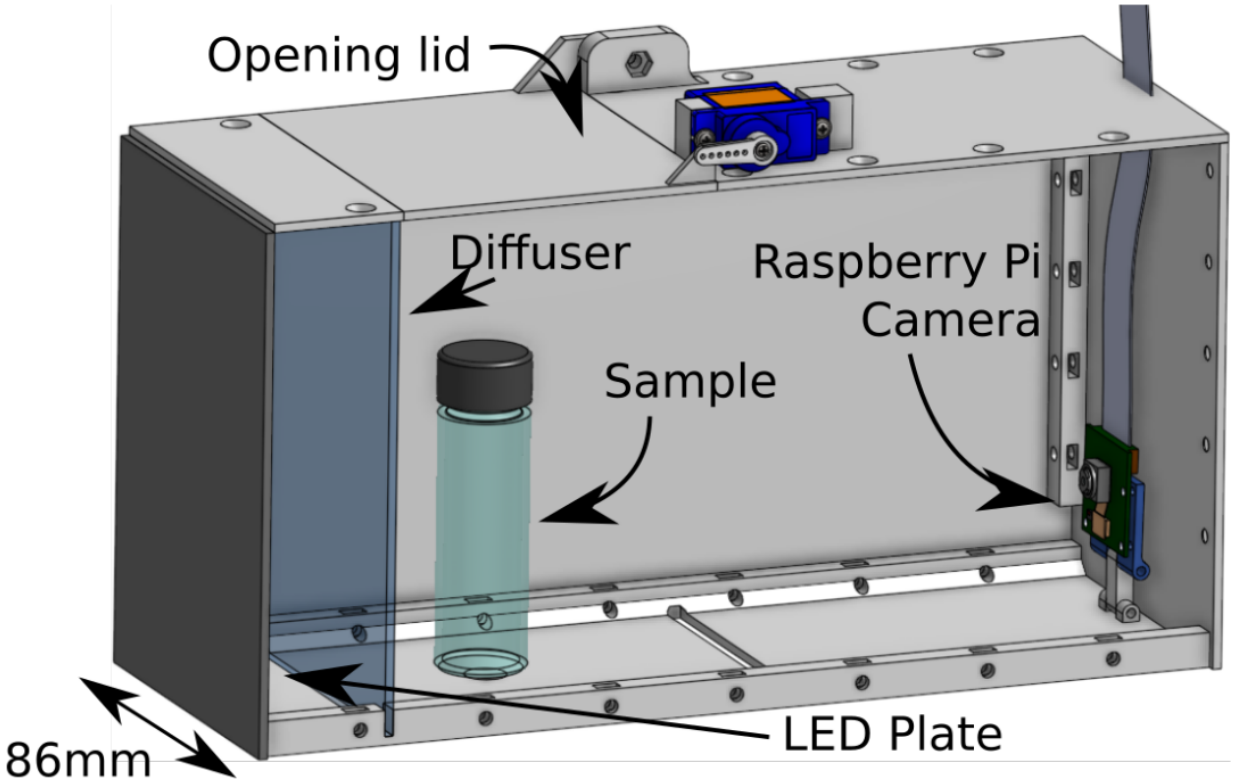


Figure 4.24: Image of the internals of the *Photo Stage* showing the major components.

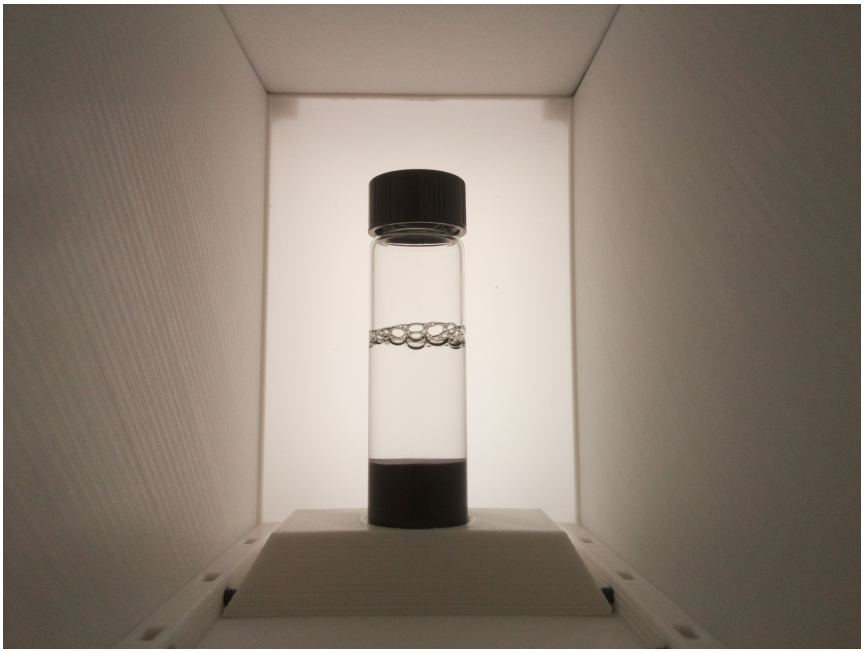


Figure 4.25: Image of a sample taken inside the *Photo Stage* using the Raspberry Pi Camera Module.

4.3.1.1 Enclosure

The *Photo Stage* is designed so that the camera is as far from the sample as possible. This reduces distortions in the image and approaches a parallel projection. However, the *Photo Stage* must not occupy more than one slot in the deck of the *OT-2*. Thus, the *Photo Stage* extends from its slot up to the wall of the *OT-2* as shown in Figure 4.26.

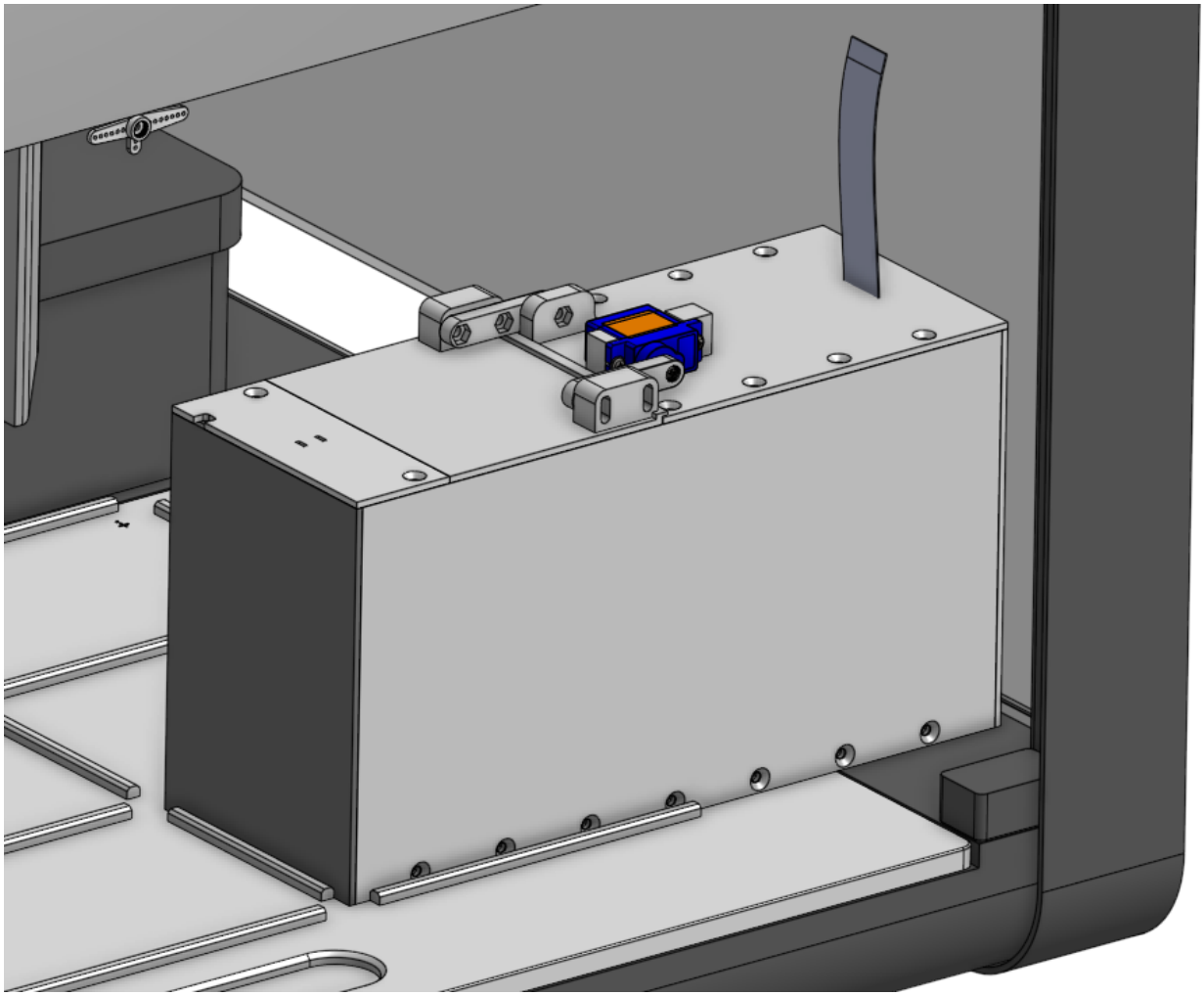


Figure 4.26: The position of the *Photo Stage* inside the *OT-2*. It extends from slot # 3 until just before the right side panel of this machine.

The enclosure is assembled from multiple 3D-printed panels joined at the edges using bolts. This is done because such flat panels can easily and quickly be 3D-printed and so that any side of the *Photo Stage* can quickly be removed to gain access to the inside. The fasteners used are countersunk M3 bolts so that the heads are flush with the panel and nuts embedded into the 3D-printed parts. The panels are printed in white PLA plastic. The white colour is used to minimize the effect the enclosure has on the light and to avoid tinting the image.

4.3.1.2 Camera

The camera used is a Raspberry Pi Camera Module V2. This model was chosen for its compact size, convenient mounting options, compatibility with the Raspberry Pi, and affordability. The camera is mounted on an adjustable arm with 3 degrees of freedom, as presented in Figure 4.27. This way, the viewing angle can be adjusted and tilt can be eliminated.

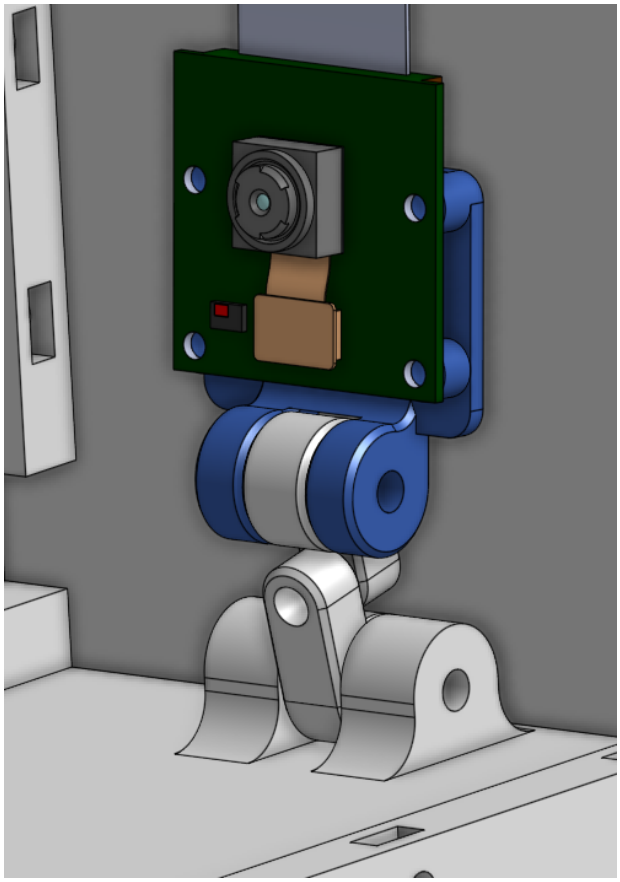


Figure 4.27: Image of the *Photo Stage* camera mount with 3 degrees of freedom.

4.3.1.3 Backlight

The LED backlight is made by adhering LED strips to an aluminium plate. The metal plate acts as a heat sink for these powerful LEDs. The lighting strips used are so-called

"high CRI" (colour rendering index) LEDs from YUJILEDS [106], meaning the intensity of light at a range of wavelengths is relatively uniform. This ensures that colours are as accurate as possible. Furthermore, it may allow for the *Photo Stage* to be used for colour characterisation (§1) instead of the spectrophotometer, an idea that has been considered during this project.

4.3.1.4 Diffuser

Between the backlight and the sample is a light diffuser. This spreads the light from the LEDs so that the surface behind the sample has uniform lighting without hotspots. The optimal distance between the backlight and the diffuser was calculated by HeeBong Yang based on the LED spacing and individual LED beam angle. The diffuser is held in place by slots embedded into the side panels and is secured in place by a small top panel. This panel is secured by only two bolts, making it easy to remove, allowing the diffuser to be slid out.

4.3.1.5 Opening Mechanism

One section of the top plate is hinged so that it can be opened to allow a vial to be deposited into or removed from the *Photo Stage*. This lid uses a small servo motor to actuate its opening and closing, the same kind of motor commonly used in model aircraft. This kind of motor is used because it is compact, because the absolute position of the shaft can easily be controlled, and because of its affordability. Two 3D-printed beams connect the lid to the shaft of the motor on one side and a passive hinge on the other side. The lid's mounting point to the beams includes slots so that the position and angle can be adjusted. This way, it is possible to ensure that the lid closes properly despite the inaccuracy of 3D printers.

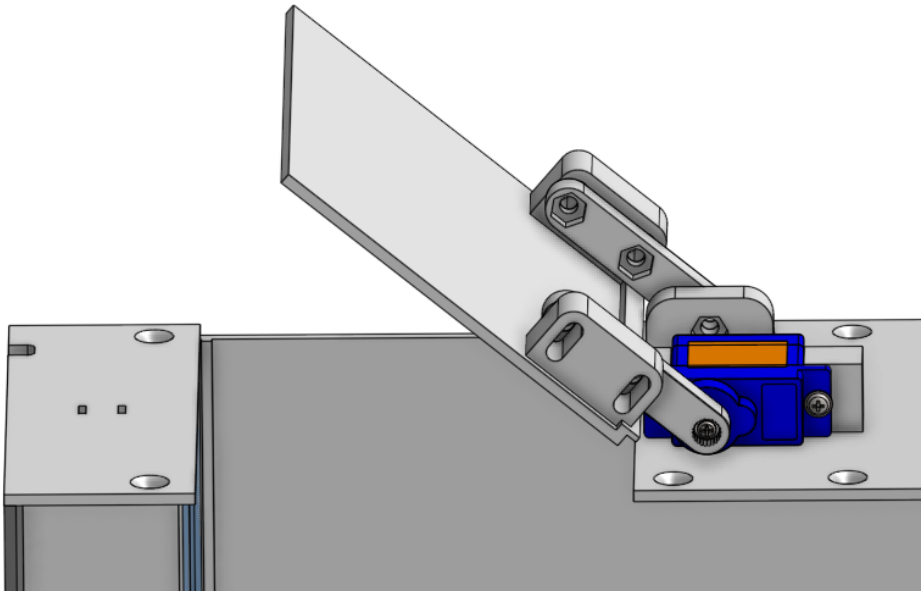


Figure 4.28: The opening mechanism of the *Photo Stage* featuring a servo motor.

4.3.1.6 Printed Circuit Board

A printed circuit board (PCB) was designed to easily manage the wiring of all of the *Photo Stage*'s electronics. Like the other custom modules, the heart of this board is the STM23F303K8 microcontroller. This microcontroller was chosen for its ease of use, because it has sufficient interfaces without being overly complicated, for its cost, and because the lab is already familiar with it in the form of the NUCLEO-F303K8 development board [96].

The PCB is powered by 24 V like all custom modules. This power can be connected and disconnected to two high power outputs. One of these is to control the backlight of the *Photo Stage*, and the other is for redundancy. These outputs can be enabled and disabled by the microcontroller via solid state relays. These act as switches, connecting and disconnecting the load from the input 24 V supply.

The board also controls the servo motor of the opening mechanism (Section 4.3.1.5). This motor needs a 6 V and ground connection for power and a Pulse-width modulation (PWM) signal to specify the desired angle. It would be convenient to power the servo motor with the 5 V power from the USB connection. Although this kind of motor can be driven at 5 V, it is not good practise to connect large loads to the USB. Therefore, a buck converter is used to step the 24 V supply voltage down to 6 V. Designing an efficient

switching regulator is difficult, so an external off-the-shelf module is included as a daughter-board. The PCB includes a sequence of eight 3-pin connectors to provide 6 V, ground, and a unique PWM signal to up to eight servo motors.

Finally, just like the other custom modules, the *Photo Stage* PCB includes a USB connector and a USB to UART converter. The USB connector provides power to the microcontroller and other low-voltage, logic-level components. The USB connector's data pins are also connected to a *FT232RL* USB to UART integrated circuit [98]. This converter enables communication between the microcontroller and the central computer. There is also a connector for the JTAG programming port.

This PCB was designed using *Autodesk Eagle* [99]. The schematic of this PCB is included in Appendix G and the board is included in Appendix B. An image of the soldered PCB is presented in Figure 4.29. The PCB was installed on the side of the *Photo Stage*, which can be seen in Figure 4.30.

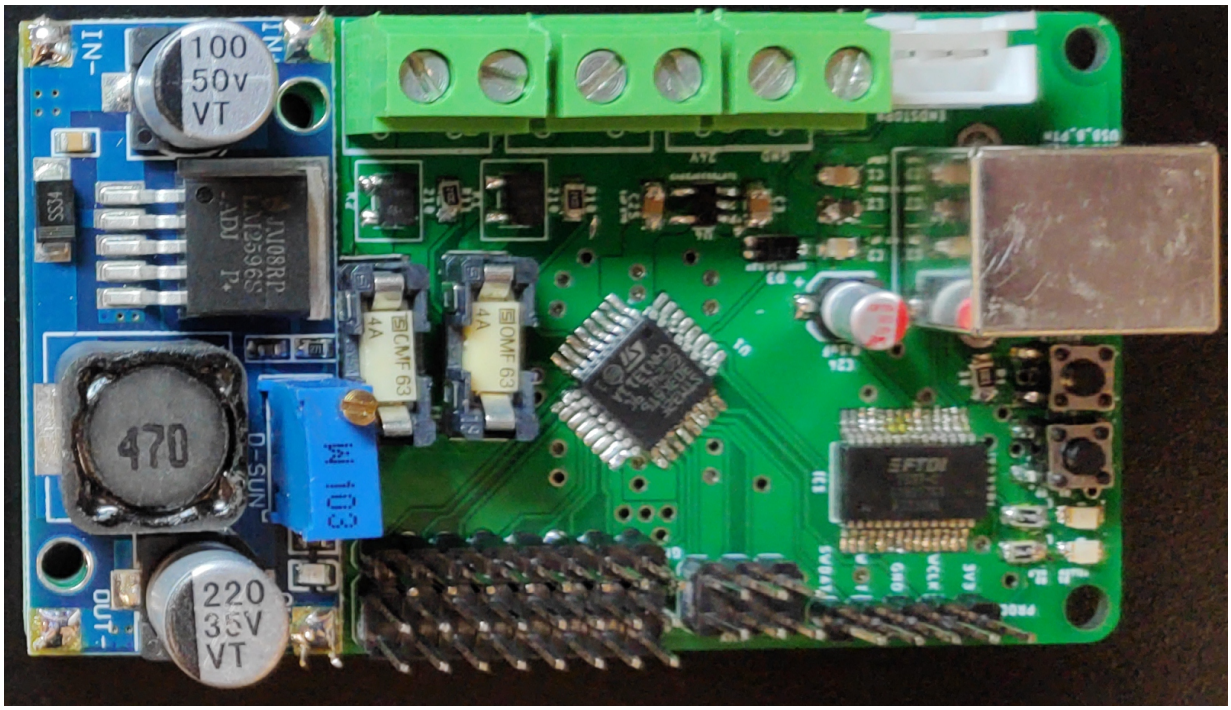


Figure 4.29: A photo of the populated PCB for the *Photo Stage*.

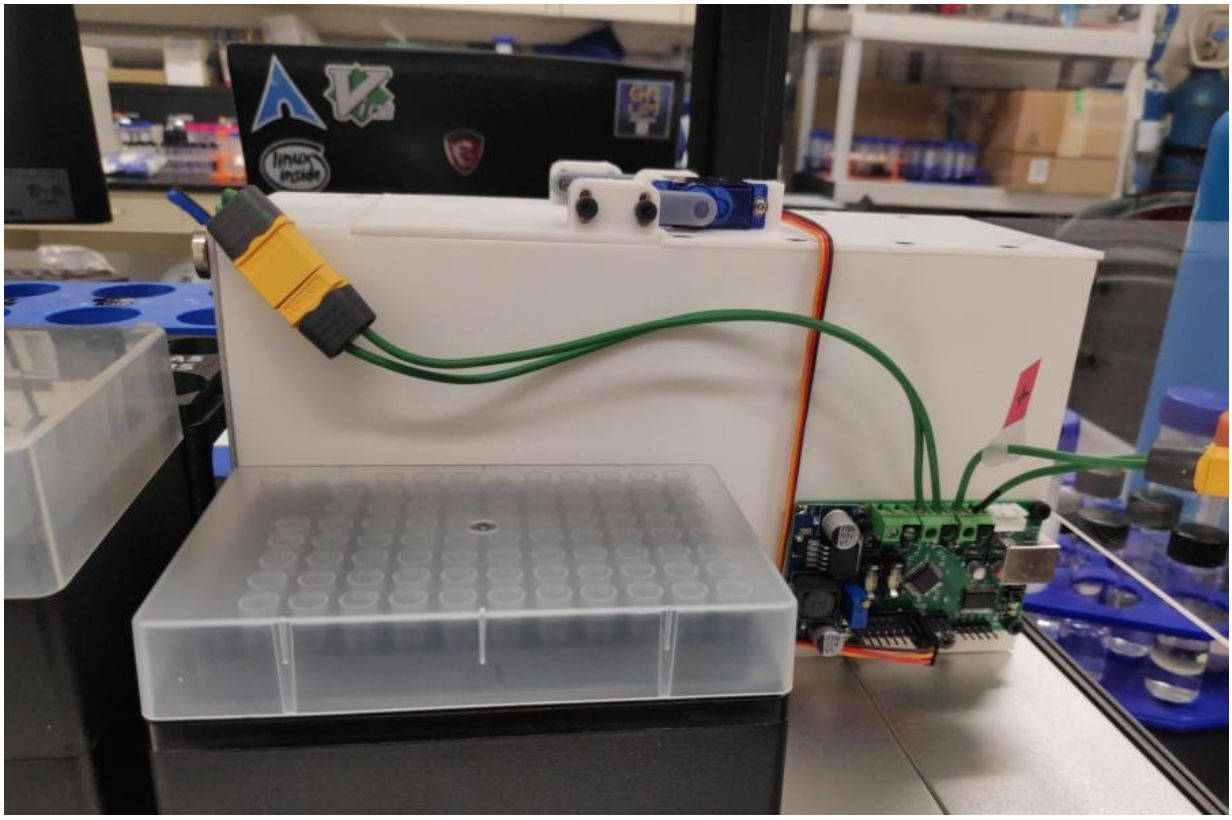


Figure 4.30: A photo of the *Photo Stage* PCB wired to the servo motor and backlight and installed on the side of the *Photo Stage*.

4.3.1.7 Software

The *Photo Stage* has two software components like all custom modules: there is firmware written in C++ using the Mbed framework [102] running on the microcontroller for the *Photo Stage* PCB and the associated Python library for use with the *Central Command* software that interacts with this firmware. The two components communicate with each other using UART and the USB connection discussed in Section 4.3.1.6. The messages follow a predefined command and response protocol (Section 4.6.1.1).

The purpose of the Python library is to simplify interacting with the firmware. It acts as a layer of abstraction. For instance, instead of manually creating the sequence of bytes that will cause the lid to open and sending this packet over the UART connection, it is possible to call the method `photo_stage.lid_open()`.

Several commands have been defined specifically for the *Photo Stage*. Each command has a code (a byte to uniquely identify the command), a list of arguments, and a response format. The communication protocol is described in detail in Section 4.6.1. The commands defined for the *Gripper* are enumerated in Table 4.5. The *Photo Stage* also supports all base commands defined in Table 4.13.

Table 4.5: Command names, codes, and functions specific to the *Photo Stage*.

Command Name	**Code**	**Function**
`WRITE_SERVO`	`0x10`	Set the position of the servo motor to a specific angle.
`LED_ON`	`0x12`	Turn on the *Photo Stage* backlight.
`LED_OFF`	`0x13`	Turn off the *Photo Stage* backlight.

The *Photo Stage* has no long-running commands, so needs no status enumeration type that the other custom modules employ.

4.3.1.8 Camera Drift Tracking

The *Photo Stage* was designed to capture repeatable images of the vial enclosed. This refers to not only consistent lighting, but also repeatable camera angles. Since this guarantee is made by the *Photo Stage*, it should be tracked and checked automatically on every image. Thus, four ArUco markers [107], [108] were generated and adhered to the diffuser plate as shown in Figure 4.31. By locating these codes in every image, it is possible to compare the expected position to the detected position to track if the camera angle is drifting over time. Furthermore, if the difference between the two is greater than a certain threshold, an error will be thrown alerting the operator that the camera has drifted too much. It may also be possible to use these markers to transform and re-align the image.

Adhering the markers to the diffuser assumes that the latter will not move relative to the vial. This is a safe assumption; both the diffuser and vial platform are securely mounted. The movement between these two should be much less than the movement of the camera, which is attached to an adjustable mount with 3 degrees of freedom (Section 4.3.1.2). Once a suitable camera angle has been found, these 3 joints should be locked to produce repeatable images. However, it is possible that this is bumped.

ArUco markers were selected because only a small amount of data needs to be stored and because this allows different markers to be used for the camera drift tracking and vial identification. Since QR codes are used for sample identification, vials can be identified by

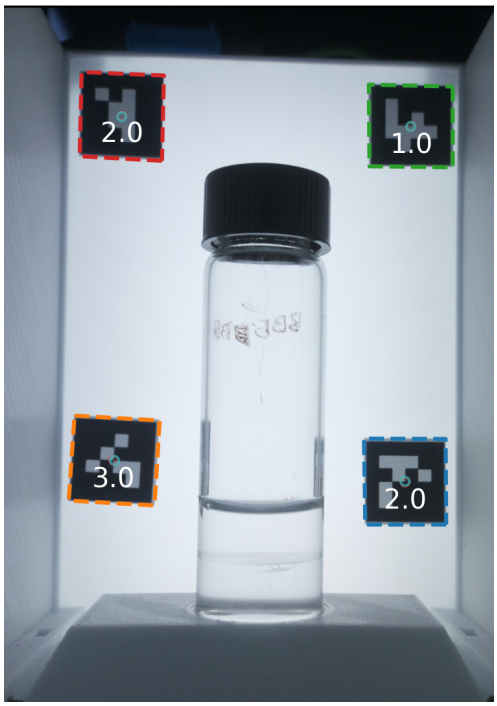


Figure 4.31: An example of the ArUco-based camera drift tracking on an image of a two-phase sample in the *Photo Stage*. Red circles show the expected centre of each marker, while blue circles show the detected centre. As the distance is small, these are overlapping. The numbers show the distance in pixels between the expected and detected centre. Such small differences are negligible for an image of this size and may be affected by variance of the ArUco detection algorithm.

searching for QR codes in the image and the markers on the diffuser located by searching for ArUco markers without conflict. Relative to the sample identifier, the markers of the diffuser do not need to store much information. Encoding the numbers 1 to 4 would be sufficient. ArUco markers are lower resolution than QR codes, so less data can be stored, but they are easier to detect in the image.

4.3.1.9 Conversion Between Volume and Height

There are many instances when using the *Photo Stage* where it is mandatory to convert between a volume of liquid and a height in a vial. For example, after the height of the boundary is detected, the volume of liquid in the top and bottom phase must be calculated in order to instruct the pipette how much liquid to extract. In order to establish a conversion between the two domains, a vial was progressively filled with water and the boundary height was carefully measured. The result of this is presented in Figure 4.32, which shows that $1\,\text{mL} = 3.81\,\text{mm}$. The y-intercept, which is a result of the slightly rounded bottom of the vial, can normally be ignored. In the aforementioned example, the top phase volume would be the difference between two heights, which is not affected by the y-intercept.

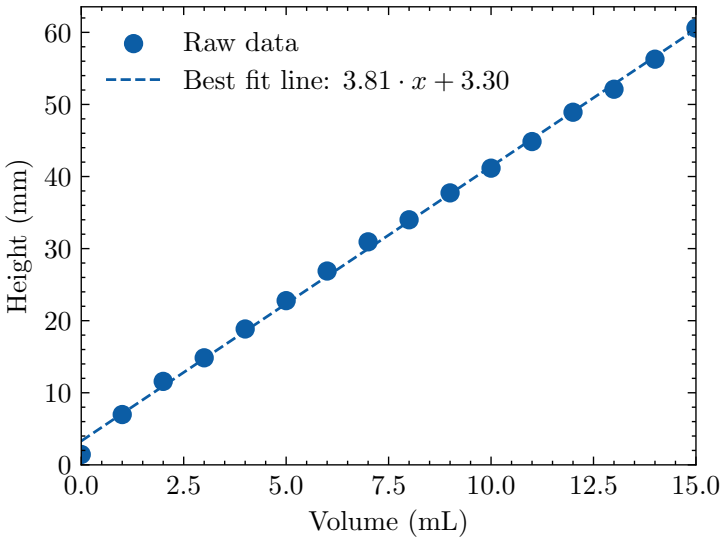


Figure 4.32: Result of the calibration of the conversion between volume and height for 14.7 mL vials used in ATPE. The mean square error of the fitting is $0.413\,\text{mm}^2$.

4.3.1.10 Conversion Between Millimetres and Pixels

Another translation that is often required with the *Photo Stage* is conversion from pixels to millimetres. The purpose of the *Photo Stage* is to recognize the phase boundary in an image of a vial. However, the direct output of the computer vision software will have the units of pixels. This must somehow be converted to millimetres so that the pipette can be instructed how high the boundary is. This is also a required prior step when converting to a volume. It would be possible to deposit an object with a known length into the *Photo Stage*, to photograph it, and to measure how many pixels it occupies in the image. However, the y-intercept is very important in this case. The required conversion needs not only to know how many pixels make up one millimetre, but also where the tip of the pipette will be if the *OT-2* is commanded to move it 0 mm from the bottom of the vial.

Since the positioning of the pipette tip is crucial when performing separation, this is used as the reference for calibration. The pipette tip was moved to 35 positions in an empty vial in the *Photo Stage* and an image was captured at each position. Then, a GUI was developed to present each image one after the other. Clicking the tip of the pipette saves the location clicked and proceeds to the next image. This is much simpler than automatically recognizing the tip in the image, and by scaling the image by a factor of four, it can be precise as well. The result of this is presented in Figure 4.33, which shows that the height y in pixels is equal to $18.83 \cdot x + 8.96$, where x is in millimetres. The zero position for this calibration is based off of how the image is cropped to the vial (Section 4.3.2.6), but the relationship can easily be updated if the cropping is changed. The calibration must be redone with new images if the camera angle changes, but this is being tracked (Section 4.3.1.8) and the calibration can be done quickly now that the code and procedure is established.

4.3.2 Separation Computer Vision

In order to achieve automatic separation, the boundary must be programmatically recognized in the image taken by the *Photo Stage*. This is the task of the separation computer vision software. Multiple attempts have been made to develop such software. Sections 4.3.2.2 to 4.3.2.5 describe the attempts made by undergraduate research assistantships (URA) students. Then, Section 4.3.2.6 presents a simpler and more reliable method that takes inspiration from the previous four. Finally, the results of this improved method are given in Section 4.3.2.7.

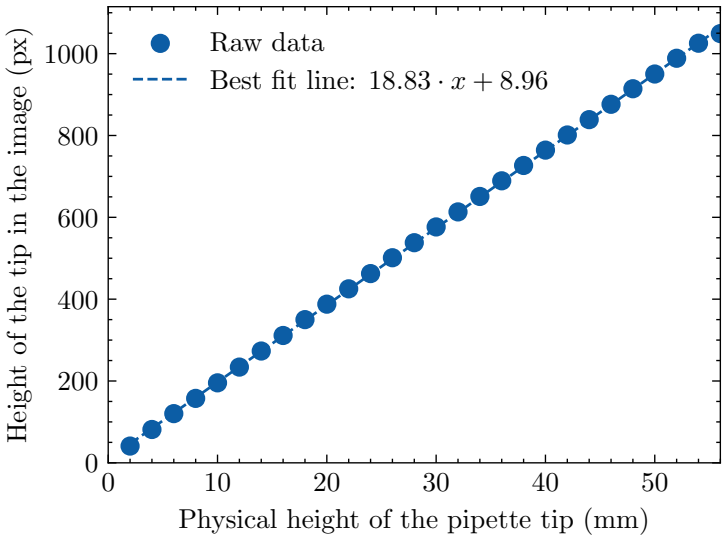


Figure 4.33: Result of the calibration of the conversion between pixels of the *Photo Stage* camera and millimetres of the *OT-2* pipette. The mean square error of the fitting is $11.52\,\mathrm{px}^2$.

4.3.2.1 Statement of Contributions

Some software described in this section was developed by Zidong Xie, Muskan Lalit, and Lu Wang. Tests of this software were performed by Zidong Xie, Muskan Lalit, Lu Wang, Luke Neal, Garen Simpson, HeeBong Yang, and the author.

4.3.2.2 Attempt # 1: Vector-LabPics Pre-Trained Models

The method devised by Zidong Xie uses three pre-trained models optimized for the recognition of materials and vessels in a chemistry lab setting [109]. The first model uses hierarchical segmentation to recognize all vessels in the image, which is used to crop the image around only the vessels, making the task of subsequent models easier. Then, the second model, a generator-evaluator-selector, isolates each vessel from the others. Finally, the third model, another generator-evaluator-selector, isolates the different liquids in each vessel. The function of each of these three models is presented in Figure 4.34. By isolating the top phase and bottom phase in this way, it is possible to determine the boundary between them [109].

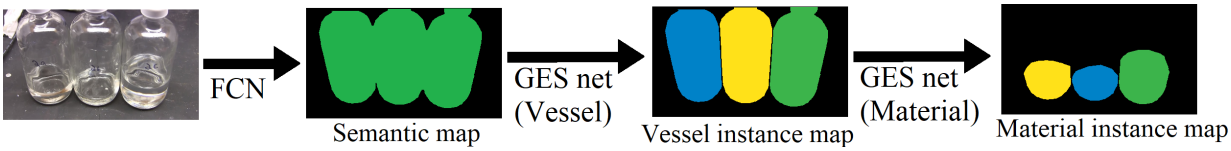


Figure 4.34: Illustration of the progression of 3 models to recognize first all vessels, then each vessel individually, then the contents of each vessel. Reproduced from reference [110] under the CC BY 4.0 license.

These pre-trained models are very performant. However, this complexity is costly. The most important problem is that the networks are very large, requiring $> 4\,\text{GB}$ of video memory to load in a desktop graphics card. It is not feasible in terms of space or time efficiency to use such large models in an embedded application. It is not possible to load the models on the Raspberry Pi of the *Photo Stage*, for example. The reason that the models are so large is that they are very general. They are able to recognize multiple different types of vessel, corks, labels, gels, vapour, and foam. This complexity is not required for this application, which always uses the same kind of vial in a fixed position in the image with consistent lighting.

4.3.2.3 Attempt # 2: MobileNet-SSD and Adaptive Thresholding

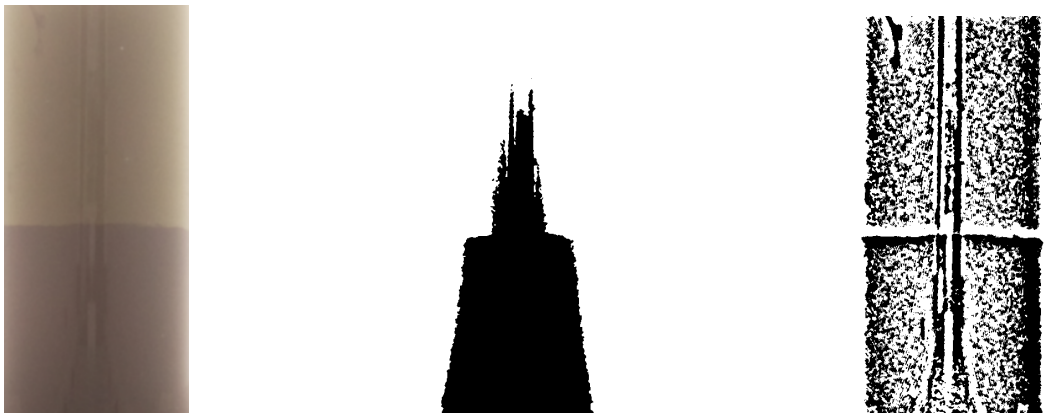
The boundary detection algorithm by Muskan Lalit first used a large publicly available pre-trained model MobileNet-SSD [111] to locate the vial in the image then used adaptive thresholding to segment the two phases. Here, SSD refers to single-shot multi-box detection, a machine learning architecture published in 2015 that detects and locates multiple objects in a single image [112]. The concept of using a pre-trained model is sound. This leverages a large and generalised network to solve the problem of locating the vial in the image.

After locating the vial in the image, the phase boundary is detected by segmenting the image using adaptive thresholding and then detecting the contours. Adaptive thresholding is similar to global thresholding, an image filter that simply colours all pixels brighter than a threshold white and all pixels dimmer than a threshold black. The difference with adaptive thresholding is that the threshold is updated from one region of the image to the next to account for changes in lighting conditions across the image [113]. Examples of the output of both methods are shown in Figure 4.35 and Figure 4.36. An example of this method on an image of a vial is presented in Figure 4.37.



(a) Input image. (b) Global thresholding. (c) Adaptive thresholding.

Figure 4.35: Visualisation of the difference between global and adaptive thresholding on an example image of an ArUco code. There is a slight change in lighting from the top left to bottom right of the image. This is represented in global thresholding. However, in adaptive thresholding, the bottom right of the image is black because the threshold changes from each corner of the image. Furthermore, the inside of the black pixels in the input image are white in the adaptive method because the threshold changes for pixels surrounded by black. Note that these parameters were chosen to show the difference between the two methods and are not optimal for either.



(a) Input image. (b) Global thresholding. (c) Adaptive thresholding.

Figure 4.36: Visualisation of the difference between global and adaptive thresholding on an exemplary image of a vial. The global method is partially successful in assigning white pixels to the top phase and black pixels to the bottom phase. Conversely, the adaptive method mostly shows noise in the top and bottom phase because the threshold changes from the top of the image to the bottom.

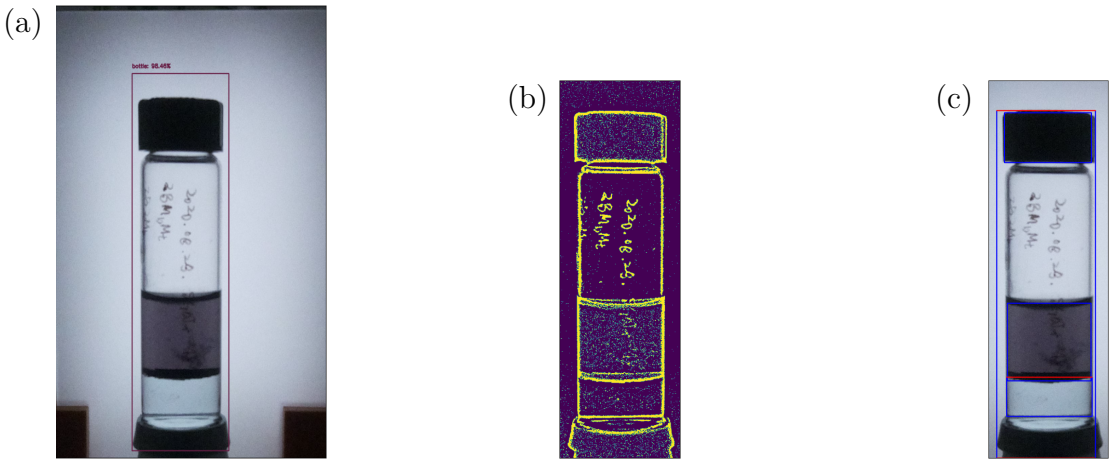


Figure 4.37: The result of Muskan Lalit's method on an exemplary image of a vial. (a) The vial is located in the image by MobileNet-SSD. (b) Adaptive thresholding is applied and contours are found. (c) The phase boundary is located. The input image was taken by HeeBong Yang and the images output from the algorithm were generated by Muskan Lalit.

When this method was tested on images of ATPE samples, the boundary was only correctly detected in 57 % of cases. Further, some of the results for examples reported to be working are questionable. Thus, this method was not pursued.

4.3.2.4 Attempt # 3: Semantic Segmentation Model

This section describes the first algorithm developed by Lu Wang. Semantic segmentation is a deep learning task whose objective is to assign a class from a finite set of classes to each pixel in an image [114], [115]. Once this is done, it is possible to know the contents of each region of the image. In this case, the classes would be the two phases. Knowing which pixels belong to each of the phases indicates the size of each phase and the height of the boundary between them.

Two semantic segmentation models are used to solve the boundary height detection task. The first separates the vial from the background, locating the vial in the image. Then, the second separates each of the two phases from the background. Knowing which pixels make up the top phase and which make up the bottom phase, it is possible to determine the height of the boundary between them.



(a) Input image.

(b) Output of the semantic segmentation method. The colours represent the class of each pixel. Output image provided by Lu Wang.

Figure 4.38: The result of the semantic segmentation boundary detection method on an exemplary image of a vial in the *Photo Stage*.

This method was ultimately rejected because of its poor performance, which can be

attributed to the small amount of training data. According to the developer of this method, Lu Wang, the semantic segmentation model would be more accurate and more reliable once a large dataset of annotated examples is built, as the results are unpredictable with the current limited dataset. Therefore, the Canny edge detection method was developed (Section 4.3.2.5).

4.3.2.5 Attempt # 4: Canny Edge Detection

This section describes the second algorithm developed by Lu Wang, which is based on Canny edge detection. This well-known edge detection algorithm detects abrupt colour changes in an image [116]. A simplified description of the algorithm is that it assigns an intensity to each pixel of the image based on its likelihood to be an edge. Then, hysteresis thresholding is used to differentiate true and false edges. The algorithm has two threshold parameters: T_{low} and T_{high}. Pixels with intensity above T_{high} are directly included in true edges, whereas those with intensity below T_{low} are discarded. Pixels with intensity between the thresholds are included based on their connectivity to true edges. That is, they are true edges if and only if they are connected to pixels with intensity above T_{high} by other such intermediate pixels.

The detected edges are typically shown in white on a black background, which can be seen in the example of Figure 4.39b. This kind of image is abbreviated as "Canny image" in the text. Since there is a visible change from one phase to the other, it is conceivable to use edge detection to find the boundary. The problem with this approach is that Canny edge detection will find every edge in the image. It is thus necessary to automatically determine which edge is the boundary between the two phases.

The image is first prepared for Canny edge detection by being converted to greyscale and blurred with a Gaussian filter, which is important given that the detector is sensitive to noise. Then, this method attempts to locate the vial in the image using Canny edge detection to detect the cap of the vial. The cap is solid black against a white background, which creates strong boundaries for the Canny algorithm to detect. Once the cap is located, it is possible to crop the image so only the vial is visible. It is also possible to crop the cap out of the image so that its edges are not mistaken for the phase boundary.

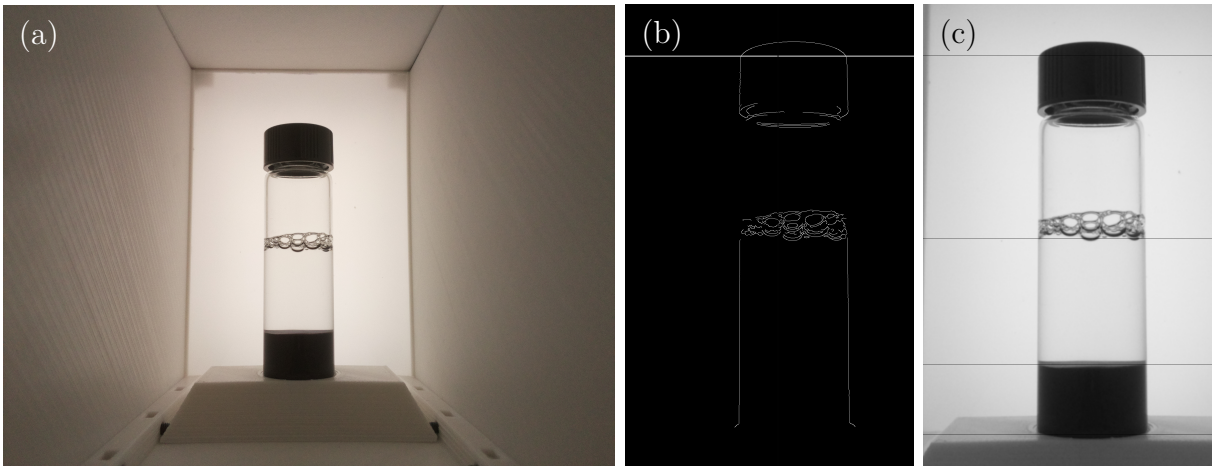


Figure 4.39: The result of the Canny edge detection boundary recognition method on an exemplary image of a vial in the *Photo Stage*. (a) The input image. (b) The result of the Canny edge detection to locate and crop to the vial (image is cropped by the software). (c) A representation of the output of the algorithm where heights are shown as horizontal lines on the cropped image. The image processing code used to make these images is by Lu Wang.

This boundary detection method was extensively tested by Lu Wang, Luke Neal, Garen Simpson, and HeeBong Yang. The most significant test, primarily performed by Luke Neal and Garen Simpson, was to use the result of the boundary detection algorithm to extract the top phase using the *OT-2*. The bottleneck with such experiments is the ATPE process itself, which was done in a hybrid fashion with certain steps being fully autonomous while others were performed manually (Section 6.1). The separation itself was also semi-automated in that photos of the sample were taken manually, the computer vision algorithm was executed manually, and the height of the detected boundary was manually inserted into a second *Opentrons* protocol. Since these early tests, the taking of the photo and image processing has been automated and the separation is handled from start to finish by a single streamlined protocol.

During the testing of this method, 16 unique samples were prepared and separated. Of these, the boundary was correctly and accurate detected in 5 cases. In an additional 5 cases, the edge representing the boundary was detected correctly, but due to other edges found in the image, this edge was incorrectly classified as the bottom of the vial or the top of the top phase. In the final 6 cases, the edge representing the boundary was completely missed. An example of each of these cases is presented in Figure 4.40. The result of this

test is a strong indication that this Canny edge detection-based algorithm is insufficient to reliably find the boundary between the two phases.

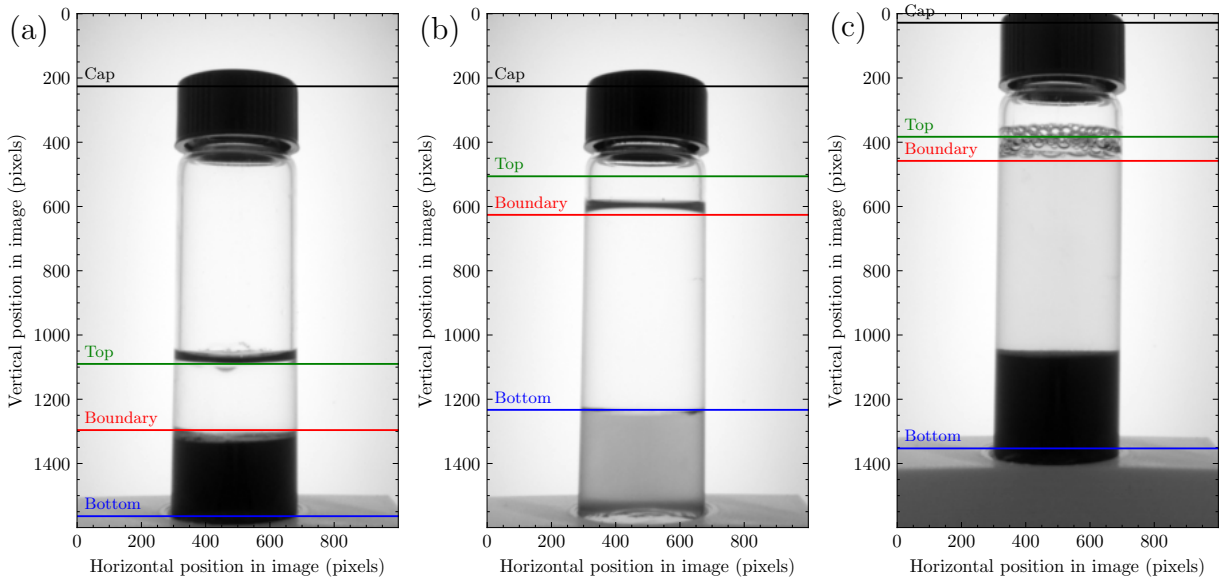


Figure 4.40: Visualisation of each possible outcome of the result of the Canny boundary detection algorithm on three exemplary samples from the separation validation test. (a) The boundary is detected correctly. (b) The boundary is mislabelled. (c) The boundary is missed. The images are cropped and the boundary data exported by software written by Lu Wang. Original images taken by Garen Simpson.

4.3.2.6 Attempt # 5: Improved Boundary Detection Algorithm

With the lessons learned from previous attempts, a more reliable boundary detection algorithm was developed by the author. It was noticed that the downfall of previous implementations is that the problem they were endeavouring to solve was more challenging and more general than required. By adding constraints that give hints to the algorithm about where to find certain features, the reliability is greatly improved. For example, a fixed cropping is used rather than locating the vial in the image. Furthermore, the theoretical liquid height in the vial is used to more reliably find the top boundary. Finally, the phase boundary search area is constrained by the detected top boundary. Thus, the steps of the improved method are (1) fixed image cropping, (2) top boundary detection, and (3) phase boundary detection. Although all of its steps will not be described until later in

this section, the flowchart of Figure 4.45 may help to understand this progression and to visualise these steps as they are described.

The *Photo Stage* was designed to capture repeatable pictures with a repeatable camera angle. Yet, all previously developed boundary detection methods first attempt to locate the vial in the image. This may be explained by sample images provided to them having variable camera angles, being taken before the *Photo Stage* was finalised, which admittedly is not ideal. The improved boundary detection algorithm foregoes this automatic detection and simply crops every image with fixed parameters. These parameters were determined by opening an example image in a photo editing program and making a rectangular selection around the region of interest. The position and size of this selection were copied into the code. This will output a repeatable region of the original image provided that the camera does not move relative to the vial in the *Photo Stage*. This condition is guaranteed by the ArUco markers on the diffuser captured by the image (Section 4.3.1.8).

Next, the boundary between the liquid and air is located in the image. The search space can be greatly reduced by using the expected boundary position as a seed value, which can be determined based on the volume of the liquid in the vial. Given that all liquid in the vial was dispensed automatically by the *OT-2*, the volume can be known and is tracked by the *Inventory System* (Section 4.5). Thus, the theoretical vial volume is converted to a height (Section 4.3.1.9), and the cropped image is further reduced to include only ~ 100 pixels above and below this theoretical height. This distance is a tuning parameter (r_{lh} in Table 4.6) that must be large enough to account for the maximum difference between the expected and actual liquid height and small enough to minimize false positives. For example, if this search space were tall enough to include the phase boundary, this may be the strongest boundary instead of the liquid height, resulting in a false positive. The boundary is detected in this image using the Canny edge detection algorithm. As before, the image is first converted to greyscale and blurred. The strongest boundary is determined by considering a rectangular region of pixels at a time, "scanning" this region from the top of the image to the bottom, and selecting the vertical position where this region includes the most white pixels. This accurately finds the boundary even if the edge is curved or if there are bubbles. An example of the latter case is shown in Figure 4.41. This method is less susceptible to outliers than the previously considered method of taking the centre of mass of all white pixels. The liquid height detection procedure is illustrated in the "Liquid Height Detection" block of Figure 4.45.

In theory, the liquid height can be calculated exactly, so this step is not required. If the volume to height conversion were perfect and the volume of the vial were tracked perfectly, then it would be possible to determine the liquid height in the image perfectly. However, there may be some error in the vial volume. The biggest source of error occurs when

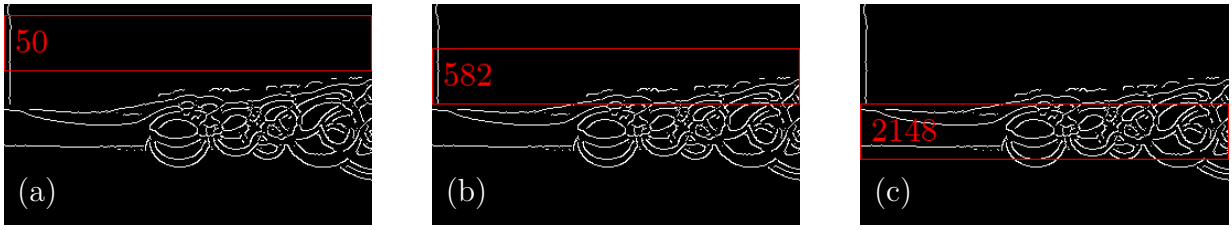


Figure 4.41: Visualisation of the method to find the strongest boundary in the Canny image, where the region being scanned is represented by a red rectangle and the number of white pixels is shown in red text. The subplots show three different vertical positions of the rectangular region. (a) Only noise is detected, resulting in a small score. (b) The tops of the bubbles are detected, resulting in a larger score. (c) The centre of the boundary and majority of the bubbles are detected, resulting in the maximal score.

the pipette accidentally aspirates air, an event that regrettably sometimes occurs during separation since the pipette tip is inserted into the top phase as little as possible so as not to contaminate the boundary. Other sources of uncertainty include error in the pipettes and liquid left in the pipette tip. Thus, the liquid height is detected, which is as much to track this volume error as it is to help find the phase boundary.

The final step is to locate the boundary between the two phases. The search space for this boundary starts just below the detected liquid height to avoid false positives resulting from the liquid-air boundary. This distance is another tuning parameter ($d_{\mathrm{lh}-b}$ in Table 4.6) which must be big enough so that any edges in the Canny image caused by the liquid-air boundary do not appear in the search space but small enough so that the phase boundary itself is inside the search area even for very small top phases. The strongest boundary is located in this search space using the same method as for the liquid height. A rectangular region is scanned down the image as shown in Figure 4.41. The phase boundary detection using Canny edge detection is illustrated by the block "Canny Phase Boundary Detection" of Figure 4.45.

This boundary detection method works well in most cases as most cases have a strong difference in colour between the top and bottom phase. However, in the early stages of ATPE, it is common to have two dark and opaque phases with little difference in colour. This is nearly impossible to detect with Canny edge detection. An attempt was made to adjust the parameters of the Canny edge detection algorithm, but increasing the sensitivity never found the phase boundary, even when false edges were detected all over the image due to noise. This is demonstrated by Figure 4.42. It may be possible to improve the detection of opaque phases by increasing the contrast or increasing the Gaussian blur, but

there is a simpler way.

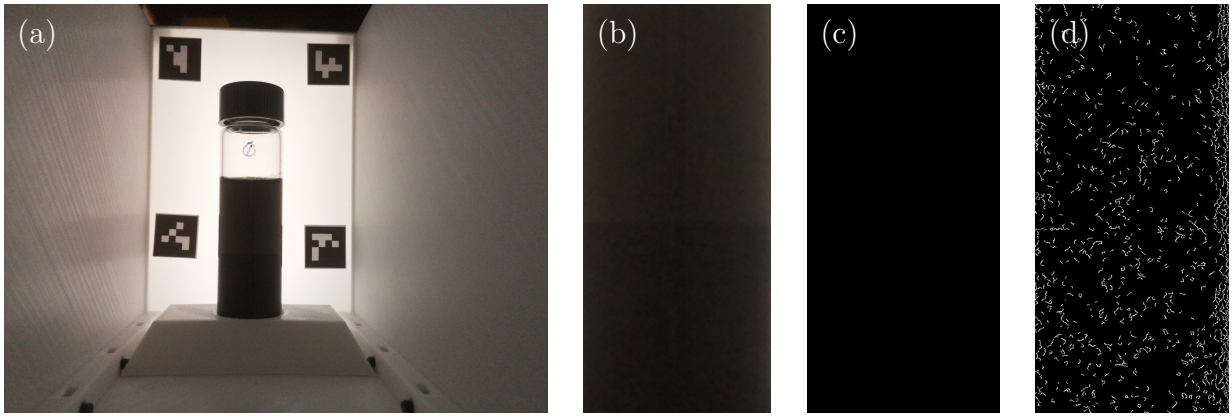


Figure 4.42: The result of Canny edge detection on an exemplary sample with two dark and opaque phases. (a) The original image of the sample in the *Photo Stage*, taken by HeeBong Yang. (b) The image cropped to the phase boundary search space. (c) The result of Canny edge detection with normal parameters $T_{\text{low}} = 20$ and $T_{\text{high}} = 50$ (Table 4.6). (d) When the sensitivity is increased to $T_{\text{low}} = 9$ and $T_{\text{high}} = 15$, the phase boundary is not detected, but edges are shown in the image resulting from noise.

When there are two opaque phases, the boundary detection task is to recognize a difference in colour vertically in the image. This is different for translucent phases, where there may be multiple edges because the boundary at the back of the vial is visible through the liquid. In this case, the Canny edge detector method is superior. For opaque phases, the vertical colour difference can be detected by compressing the image into one dimension, creating an image with a resolution of $1 \times N$ where each row of pixels in the original image is averaged into a single pixel. By differentiating this compressed image, it is possible to accurately detect the sharpest colour change, even if the two phases have similar colours. Averaging the colours in this manor is comparable to extreme blurring in one direction, so this method is robust to noise. This method, which will be referred to as the compression method in the text, is demonstrated in Figure 4.43.

Before the development of the compression method, an attempt was made to solve the problem of two opaque and like-coloured phases with image thresholding, a method similar to Muskan's (Section 4.3.2.3). Adaptive thresholding, the algorithm proposed by Muskan, was attempted first. However, "adaptive" here means that thresholding will be performed relative to surrounding pixels. This is suited for local thresholding, but here the global separation of two large phases is desired, so adaptive thresholding does not

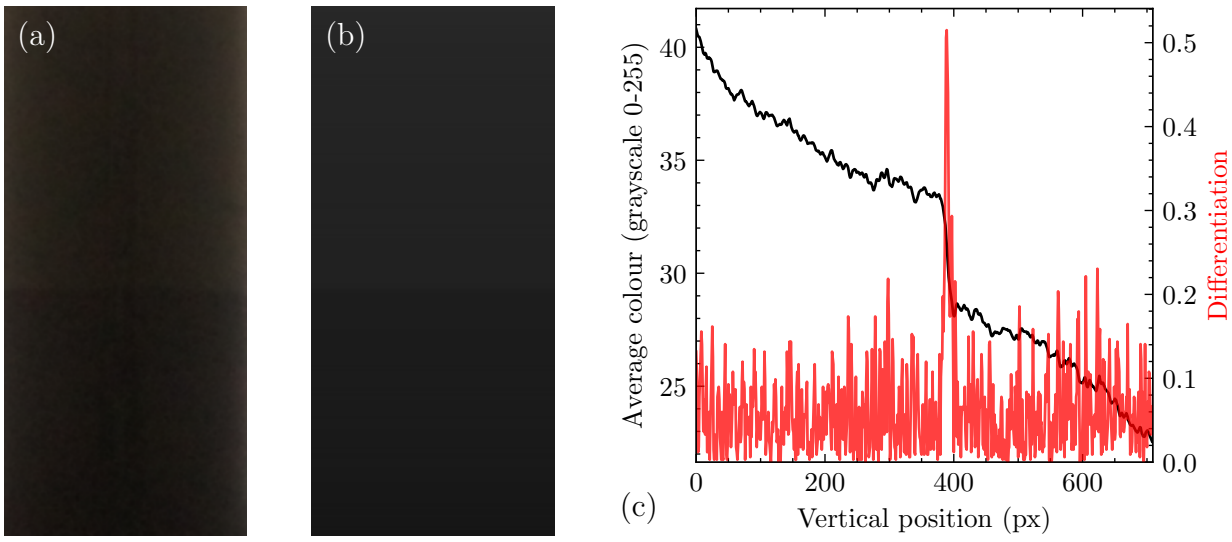


Figure 4.43: The result of the compression colour averaging method on an exemplary sample with two dark and opaque phases. The original input image is Figure 4.44a. (a) The image cropped to the phase boundary search space. (b) The averaged image where each row of pixels has the average colour of the same row of (a). This image has the same resolution as (a) so that it can be seen, but internally in the algorithm the image has resolution $1 \times N$. (c) A plot of the 1-dimensional image (black) and the absolute value of its differentiation (red), which shows the difference in colour from one row of pixels to the next. The maximum is taken as the boundary.

work. Thus, global thresholding was used. A single colour is selected as the threshold, and pixels brighter than this are assigned pure white while pixels dimmer are assigned pure black [113]. Ideally, all the pixels of the top phase would be assigned pure white and all the pixels of the bottom phase would be assigned pure black (the inverse would also be acceptable). However, due to bleed of the backlight, the edges of the vial are assigned white. Furthermore, due to reflections in the centre of the vial, which points towards the camera, part of the top phase is coloured black. Thus, only a single column of pixels was considered between these two anomalies. It was possible to detect the phase boundary in one exemplary image, which is presented in Figure 4.44, but it was necessary to greatly increase the exposure time. This method is sensitive to the shutter speed and the colour of each phase, so it was ultimately abandoned for the compression method.

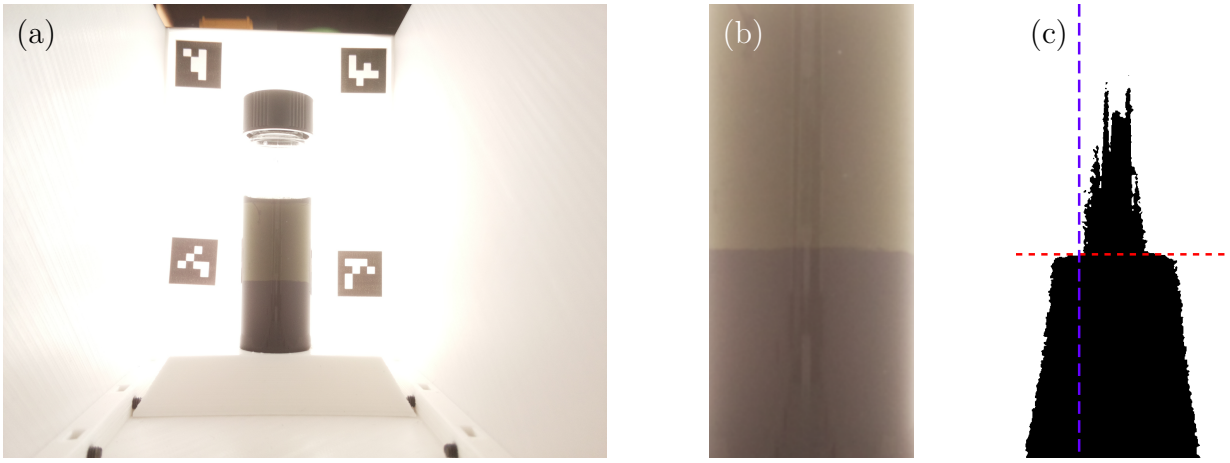


Figure 4.44: The result of thresholding edge detection on an exemplary sample with two dark and opaque phases. (a) The original image of the sample in the *Photo Stage*, taken by HeeBong Yang. This is the same sample as Figure 4.44a, but with a much longer exposure of 1000 µs as opposed to the default of 200 µs. (b) The image cropped to the phase boundary search space. (c) The result of global thresholding on the image, and the boundary (red) detected by considering a single column of pixels between anomalies (purple).

The compression method was tested for all phase boundary images, not only those exhibiting two opaque phases. However, the results were not exact. Although the detected boundary was usually near the true boundary, there were often slight errors and variance between images of the same sample with different shutter speeds for samples with transparent phases. Indeed, only 71 % images from the dataset are classified as "OK", 19 % are "not exact", and 10 % are "not good". Thus, the compression method is only used if the

Canny edge detector fails. The cases where the compression method is advantageous are those with opaque phases having little contrast. The Canny edge detector will output very few white pixels in these cases (there will still be some edges caused by noise, vial edges, etc.). Thus, if the number of white pixels in the Canny image is below a certain threshold (Q_{ms} in Table 4.6, where "ms" is for "method switch"), algorithm with automatically switch to the compression method. The progression of all steps discussed up until this point, including automatically switching to the compression method, is illustrated by the flowchart of Figure 4.45.

Table 4.6: Tuning parameters of the improved phase boundary detection algorithm.

Name	**Value**	**Description**
r_{lh}	100	The "radius" of the search space to locate the liquid height, centered around the theoretical value.
$d_{\text{lh}-b}$	50	The distance between the detected liquid height and the top of the phase boundary search space.
Q_{ms}	95	The number of white pixels in the phase boundary Canny image under which the alternative method is used.
T_{low}	20	Low Canny hysteresis threshold [116].
T_{high}	50	High Canny hysteresis threshold [116].

4.3.2.7 Experiments and Results

The improved boundary detection algorithm presented in the previous section works reliably and is being used in hybrid ATPE (Section 6.1). Of the 33 ATPE samples that have have been tested so far, only one received the classification "not exact" for the boundary detection. All other samples received the classification "OK". The challenging case of two opaque and similarly coloured phases is now detected reliably. The only situation that is still troublesome is two transparent phases of similar colour. This difficult sample is presented in Figure 4.46. Similarly, only one image was classified as "not exact" for the liquid height detection. However, in this case, there was writing on the vial which caused false edges in the Canny image. This text will eventually be replaced by QR codes (Section 4.5). While the QR code may cover part of the image, it can be recognized and removed.

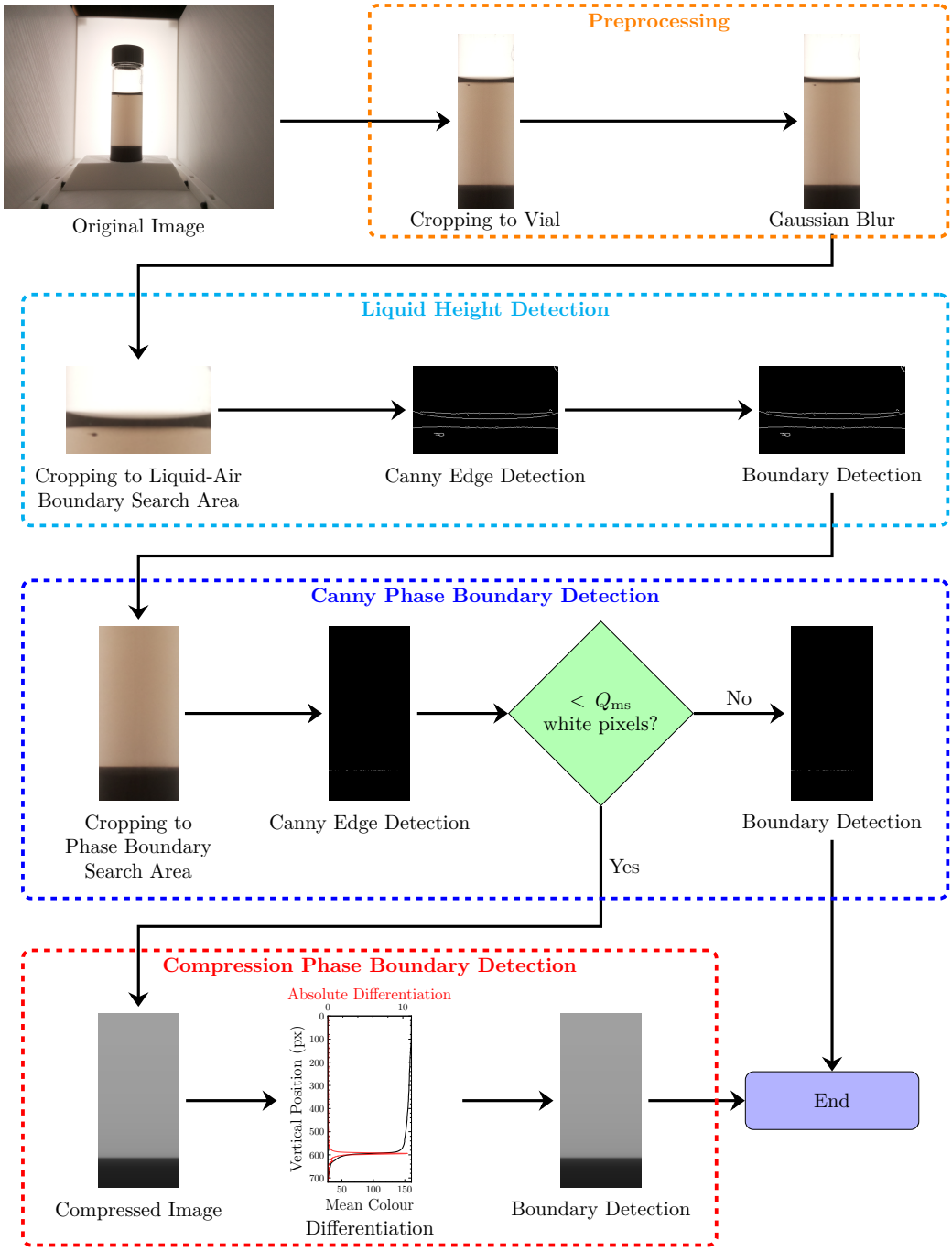


Figure 4.45: Flowchart showing the order of steps in the improved boundary detection algorithm and the automatic changing of methods if the Canny detection fails. An image that works well in both methods was chosen for the demonstration. The input image was taken by HeeBong Yang.

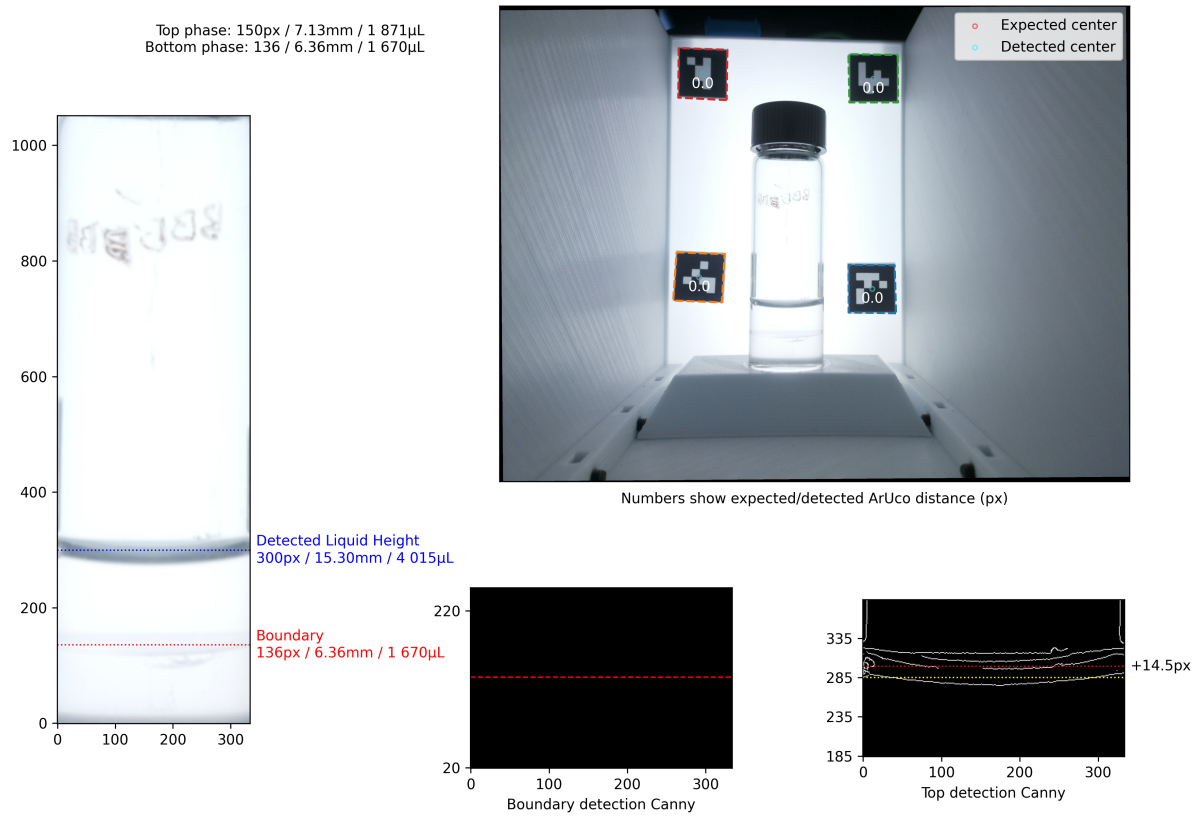


Figure 4.46: Validation image for the only remaining incorrectly detected two-phase system with two very clear phases. The validation image is the image shown to the operator for confirmation when performing hybrid ATPE (Section 6.1).

Automatic extraction of the top phase was attempted based on the output of the computer vision software. The results of this are promising, yet imperfect. It was possible to extract much of the top phase and deposit it into an empty vial without contaminating either phase with the other, which is an encouraging result. However, a large buffer of liquid must be left in the top phase because if the *OT-2*'s pipette comes too close to the phase boundary, it will become contaminated, aspirating some liquid from the bottom phase. Note that during manual and automated separation, a small amount of liquid on either side of the boundary is discarded. This amount should be minimized in order to maximize yield. However, the volume of the top phase left behind in the automated method is still

greater than that of the manual method. In fact, much liquid remains around the edge of the vial, wicking up the glass walls by capillary action as visible in Figure 4.47. This phenomenon, referred to as concave meniscus, is the result of surface tension [117].

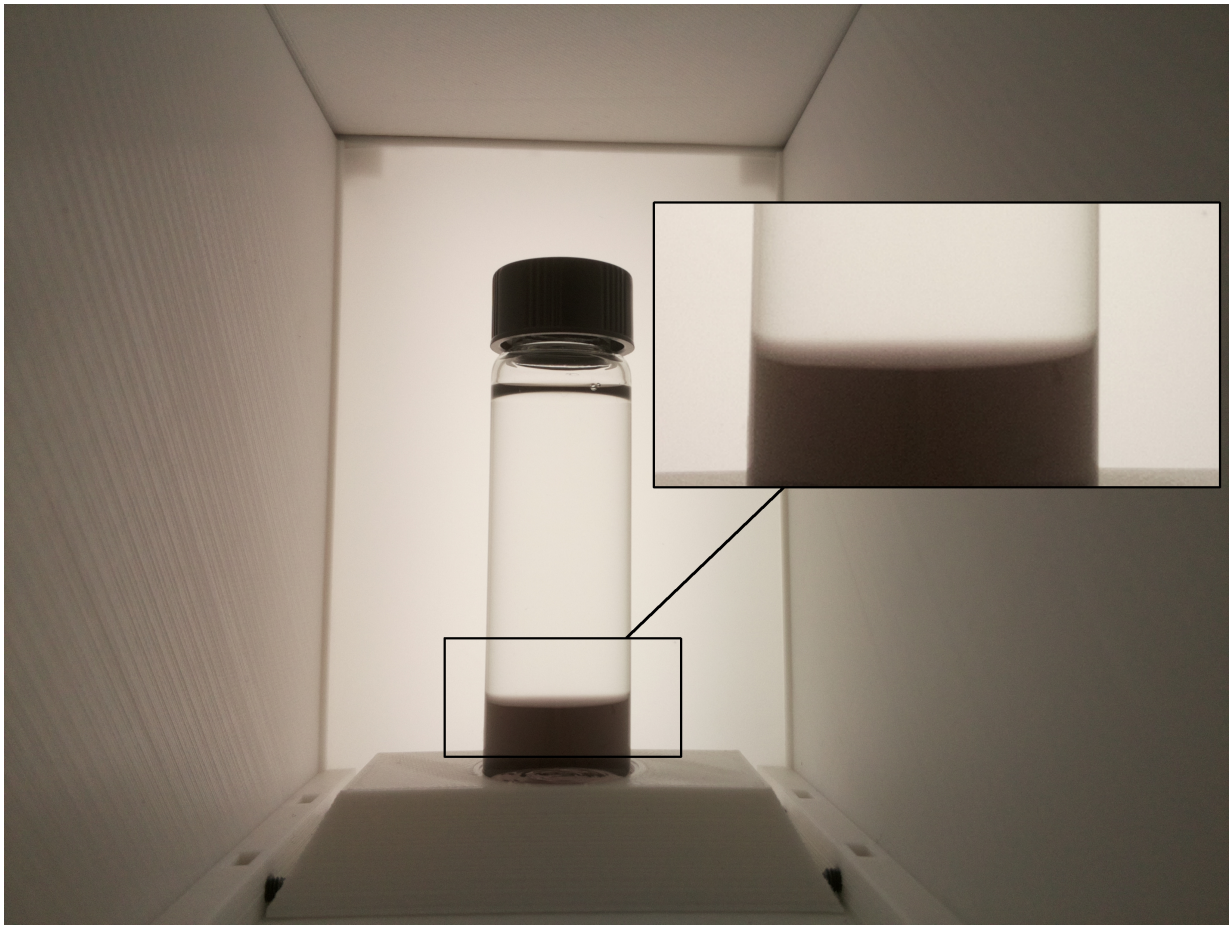


Figure 4.47: Closeup of the meniscus between the two phases for a sample in the *Photo Stage*. It is possible to notice that the boundary is concave towards the top with the liquid wicking up the sides of the vial by capillary action. Original image taken by Garen Simpson.

This meniscus issue underlines a major limitation of the *OT-2*: the pipettes are always positioned vertically, parallel to the major axis of the vial. In order to extract the small remaining volume when performing separation manually, the pipette is held at an angle and rotated around the edge of the vial in order to aspirate the meniscus. The plan

for remedying this issue is to use the external robotic arm (Chapter 5) or some device inside the *OT-2* to hold the vial at an angle under the pipette of the *OT-2* and to move and rotate it mimicking the vial-pipette interaction used during manual ATPE. This will be an important aspect of the next phase of this project. Still, the initial result that nearly all of the top phase could repeatably be extracted in an automated fashion by the *OT-2* is extremely promising. The separation of the two phases is the most tedious, time-consuming, and error-prone aspect of manual ATPE. This is one of the most significant tasks of ATPE to automate.

4.4 *Powder Handling System*

As discussed in Section 2.3.2, which introduced ATPE and presented the theory behind this process, several chemicals in liquid form are required. Specifically, the required stocks are polyethylene glycol (PEG), dextran (DX), sodium dodecyl sulfate (SDS), sodium deoxycholate (DOC), and sodium cholate (SC). While the preparation of these is not part of the main ATPE process, it is laborious and should be automated to increase repeatability. However, being outside of the main ATPE process, its automation has a lower priority.

Each of these chemicals is purchased in a powdered form and must be precisely mixed with deionized water to achieve a specific concentration. The stocks and required concentrations are presented in Table 2.1. The polymers, PEG and DX, are required in much larger quantities than the surfactants, as this table shows, so can efficiently be made manually in large batches. To make these stocks in the *OT-2* would require over thirty transfers as the largest pipette for the *OT-2* is 1000 µL. Thus, the only stocks to manufacture automatically are SDS, DOC, and SC.

4.4.1 Plan

The *OT-2* already provides a platform for precise liquid transfer, so the task of adding the deionized water is solved. Yet, a powder dispenser is lacking. Since powder handling is extremely difficult [118], which is corroborated by the many attempts presented in the following sections, it is advantageous to relax the precision on the powder weight. Indeed, after an imperfect amount of powder has been dispensed, the tube can be weighed with high accuracy. At this point, the appropriate amount of water can be added by the *OT-2* to precisely achieve the target concentration. After all, the concentration is the metric that matters; the volume can vary.

Thus, an accurate balance and powder dispenser are required, where inaccuracies in the latter are tolerable and will be corrected. There must be at least three dispensers, one each for SDS, DOC, and SC. Space should be left for two redundant dispensers for any future expansion. Furthermore, a system is required to move the empty tubes to the powder dispenser. This system would also serve the additional purpose of tube storage. Space inside the *OT-2* is limited, so this tube management system should have a large capacity of empty and full tubes. This way, tube racks are not required, saving precious deck slots. The balance and dispenser are discussed in Section 4.4.4 and Section 4.4.2, respectively. Finally, the tube management system is described in Section 4.4.3.

4.4.2 Powder Dispenser

The powder dispenser, as stated in Section 4.4.1, is a component of the *Powder Handling System* that must be able to dispense powdered chemicals with different properties relatively accurately. Inaccuracy is tolerable as the true amount dispensed will be weighed precisely and error will be compensated for by dispensing a different volume of water with the *OT-2*. The dispenser must be relatively small in size as four copies are included inside the *OT-2* for the different chemicals used in the ATPE process.

It is commonly said that automated powder handling is difficult [119], and indeed that was witnessed and corroborated during the multiple attempts in the following sections. Three designs were attempted by the author, while two others were carried out by a student in the undergraduate research assistantships (URA) program: Ardhendu Shekhar Bhattacharya. The designs for this student are based on a pneumatic system using pressurized air to push the powder and a gravity-fed iris valve. These are not the work of the author and are not detailed in this thesis. These designs are mentioned only to indicate that five powder dispenser designs have been attempted in total.

4.4.2.1 Attempt # 1: Sliding Powder Dispenser

The first attempted powder dispenser design featured a sliding mechanism that would shuffle the powder from a source tube full of powder to an empty destination tube. Since the same volume of solution with the same concentration of the chemical is made every time, the volume of the powder to dispense for a given chemical is consistent. Hence, a cavity of a fixed size can be used to transport the same volume of powder like a measuring cup with a volume designed specifically for a particular process. The operating principle is illustrated in the cross-sectional view of Figure 4.48a. This simple design was attempted

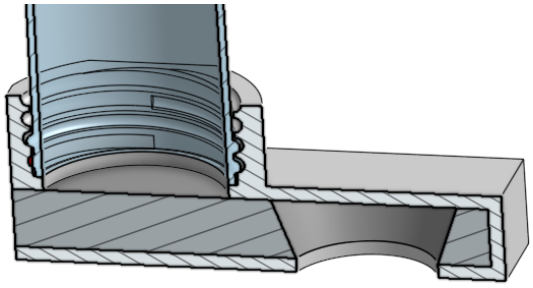
because it was thought to be manufacturable with the processes known to be available at the time, namely 3D printing. Alternatives like an Archimedes screw (Section 4.4.2.2) were considered, but the screw was deemed too advanced to fabricate.

§1 Prototype The first prototype of this concept was designed using four thin panels as the walls and a large cuboid slider with a cavity to allow a fixed amount of volume to pass. This was only a proof of concept that was actuated manually, so no actuator or mounting mechanism was included. The threads of the tube were designed into the top and bottom plate, allowing them to be securely attached. An image of this prototype is presented in Figure 4.48. All of these components were printed on an FDM 3D printer. The pieces were assembled into the dispenser shown in Figure 4.48c.

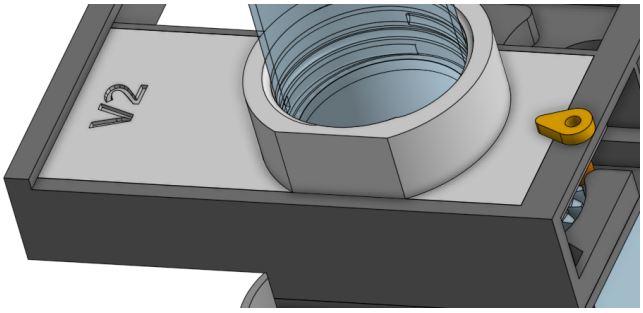
The prototype was tested with the assistance of HeeBong Yang. The dispenser was actuated by hand with PEG powder in the inverted source tube. Although the dispensers are not intended to dispense PEG due to the large volume requirements, this powder is convenient for testing due to its relative affordability and long shelf life. The results of this experiment were promising. The dispenser was able to transfer powder from the source tube to the destination tube. Some powder was left behind in the mechanism. However, since the volume of the water will be adjusted accordingly, this is not a concern.

§2 Refined version With this positive result from the prototype, a more refined design was implemented. At this point, the lab came into possession of a masked stereolithography (MSLA) resin 3D printer, allowing prints with much higher resolutions. This enabled the dispenser to be printed with much smoother sliding surfaces compared to the bumpy walls observed from filament 3D printers. Furthermore, the housing of the dispenser could be printed in a single piece thanks to the different support material requirements of MSLA printers, removing the need for fasteners. This has the benefit of reducing each dispenser's width, meaning that four dispensers can be mounted on one side of the *Conveyor* (Section 4.4.3). A cross-sectional view of this new dispenser is presented in Figure 4.49a.

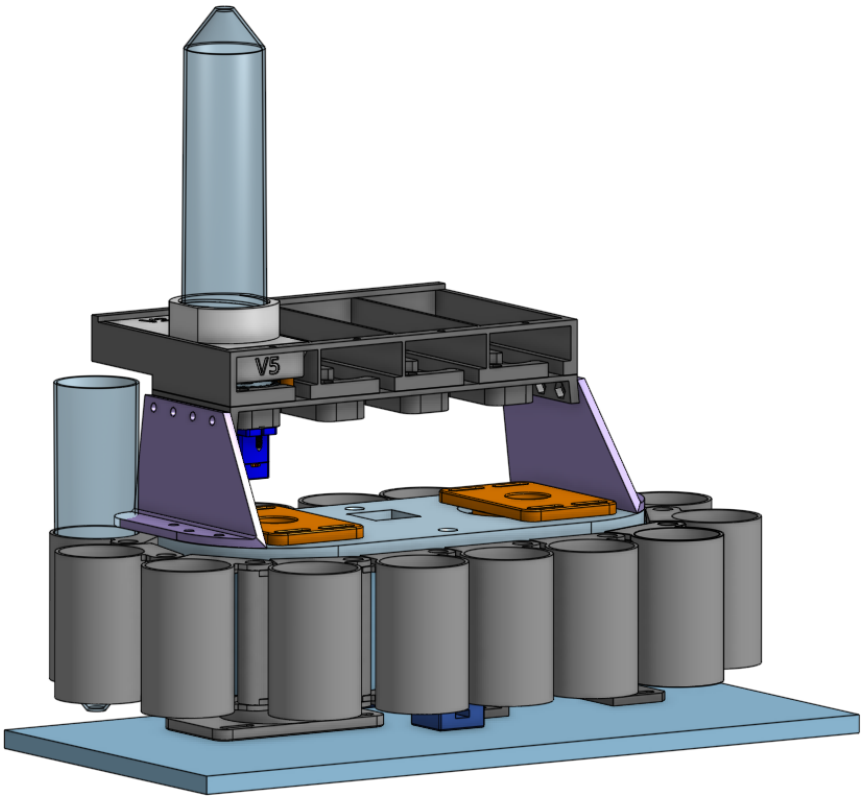
The updated dispenser is designed for integration and not as a prototype. Thus, a mounting mechanism and an actuator are implemented. The sliding dispenser itself is kept modular so that this subassembly can be removed from the larger system to refill the source tube with powder. The source tube is inverted so that the powder is fed by gravity to the dispenser. Hence, the tube must be threaded into this mechanism before it is inverted so that the powder does not spill during this process. This is simple if the design is modular and a single dispenser can be removed and refilled individually.



(a) Cross-sectional view showing the unified enclosure.



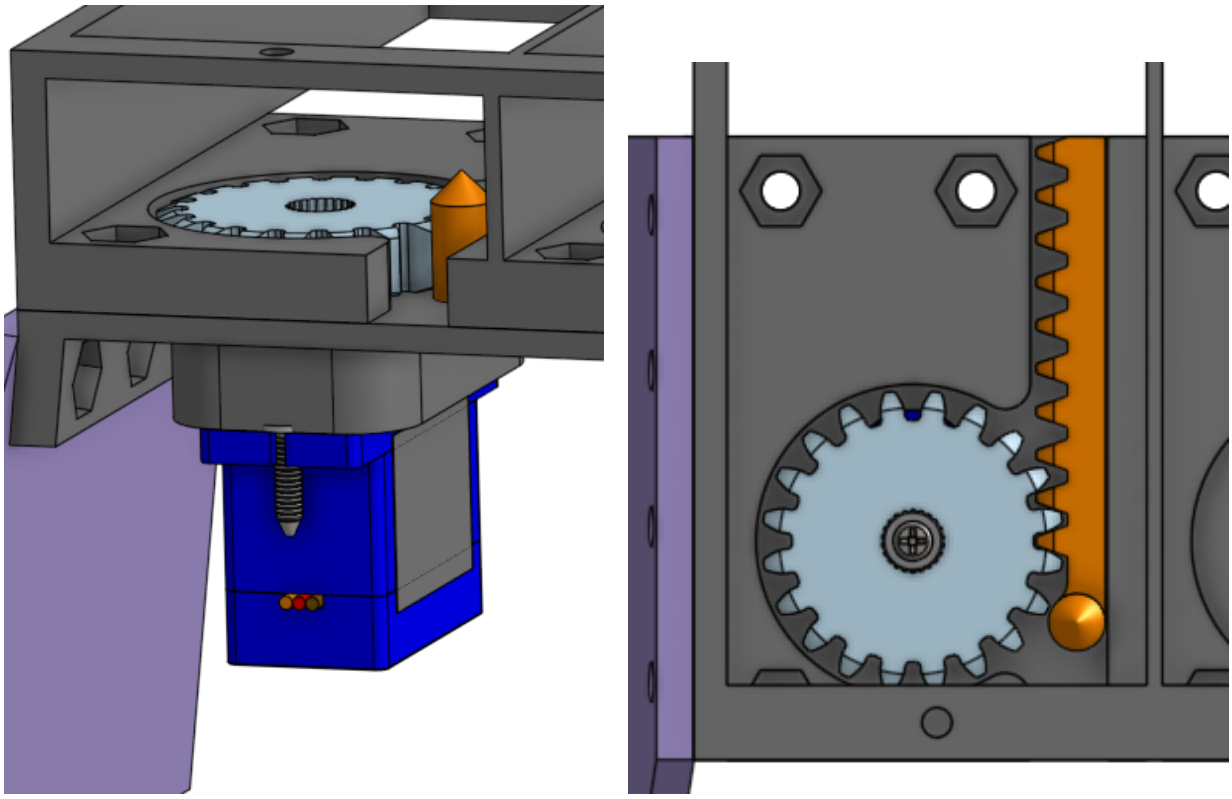
(b) The latching mechanism to hold the dispenser in its mount. The small orange piece is rotated by 90° to free the dispenser. The opposing end is constrained by a small overhang in the mount (dark gray piece).



(c) The dispenser mount supporting one of four dispensers above the destination tubes in the *Conveyor* (Section 4.4.3).

Figure 4.49: CAD screenshots of the refined sliding powder dispenser.

This dispenser subassembly latches into a mount that supports it over the destination tubes (Section 4.4.3). The mount is presented in Figure 4.49c and the latching mechanism in Figure 4.49b. The mount has the capacity to support four dispensers. A mechanism to actuate the sliding action is also included. This was designed to seamlessly engage with the dispenser as the latter is inserted into the mount. The slider of the dispenser mates with a pin on a linear gear (orange in Figure 4.50). This linear gear meshes with a circular gear, which in turn is driven by a servo motor. The servo motor, presented in Figure 4.50a, is identical to that of the *Photo Stage* (Section 4.3.1). This model is used for its compact size, absolute position control, and affordability. The two gears are 3D-printed using the MSLA resin printer, while the remainder of the mount is printed in filament.



(a) View of the actuator showing the servo motor attached underneath the dispenser mount.

(b) Top-down view of the gear system used to actuate the sliding motion.

Figure 4.50: CAD screenshots of the sliding dispenser's actuating mechanism.

§3 Electronics A PCB was created for this powder dispenser design. The board facilitates the connection of the servo motors used to actuate each dispenser subassembly and of the stepper motor used to actuate the *Conveyor* (Section 4.4.3). The primary component of this PCB is the microcontroller. Yet again, the STM23F303K8 [96] is chosen for consistency between custom modules.

Similarly to the *Photo Stage*'s PCB, this board has a switching regulator to supply the servo motors with their required 6 V. The signal wires of these motors are connected to PWM pins of the microcontroller. The circuit board also includes a stepper motor driver to actuate the *Conveyor* and a connector for an end stop to detect the zero position of the *Conveyor* (Section 4.4.3). The motor driver used is the *TMC2209* from *Trinamic* [97] like all other custom modules using stepper motors.

Finally, the PCB includes a USB connector and a USB to UART converter. The USB connector provides power to the microcontroller and other low-voltage, logic-level components. The USB connector's data pins are also connected to a *FT232RL* USB to UART integrated circuit [98]. This converter enables communication between the microcontroller and the central computer.

This PCB was designed using *Autodesk Eagle* [99]. The schematic of this PCB is included in Appendix I and the board is included in Appendix J. An image of the soldered PCB is presented in Figure 4.51.

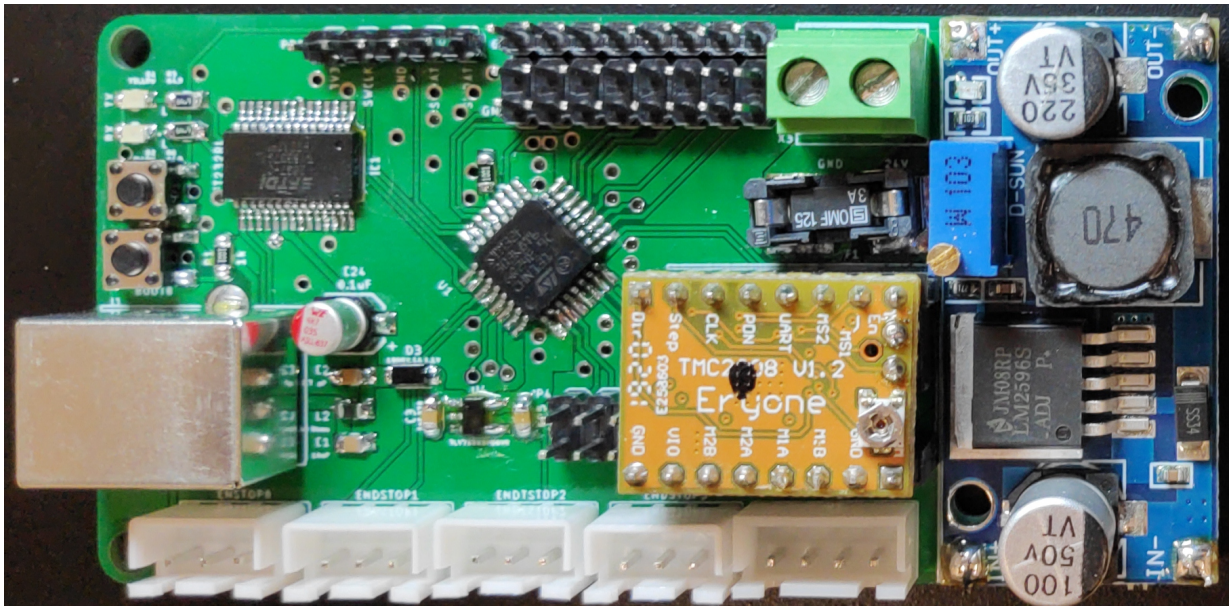
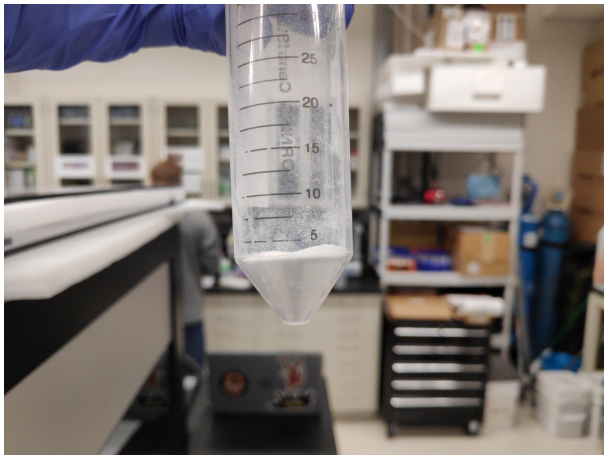


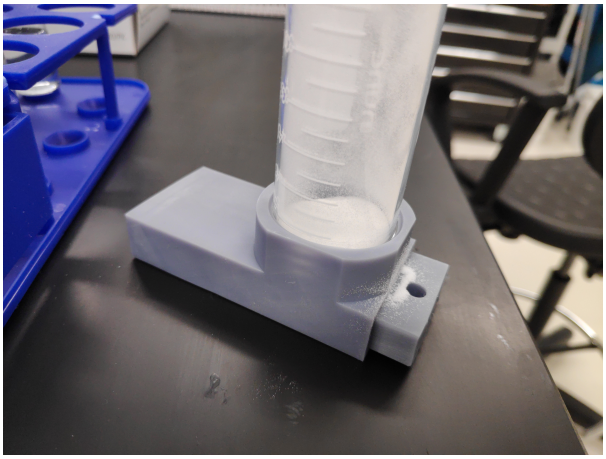
Figure 4.51: A photo of the populated PCB for the *Powder Handling Sliding Dispenser*.

§4 Software

§5 Experiments and Results This refined sliding powder dispenser printed in resin was first tested with PEG powder, which worked well. However, during the subsequent tests with the much finer SC powder, a chemical that will truly be used with the dispenser unlike PEG, this design was deemed a failure. While it was possible to dispense approximately the correct amount of powder, as seen in Figure 4.52a, the dispenser leaked badly. As presented in Figure 4.52b, an alarming amount of powder remains on the top of the slider even after actuating the mechanism a single time.



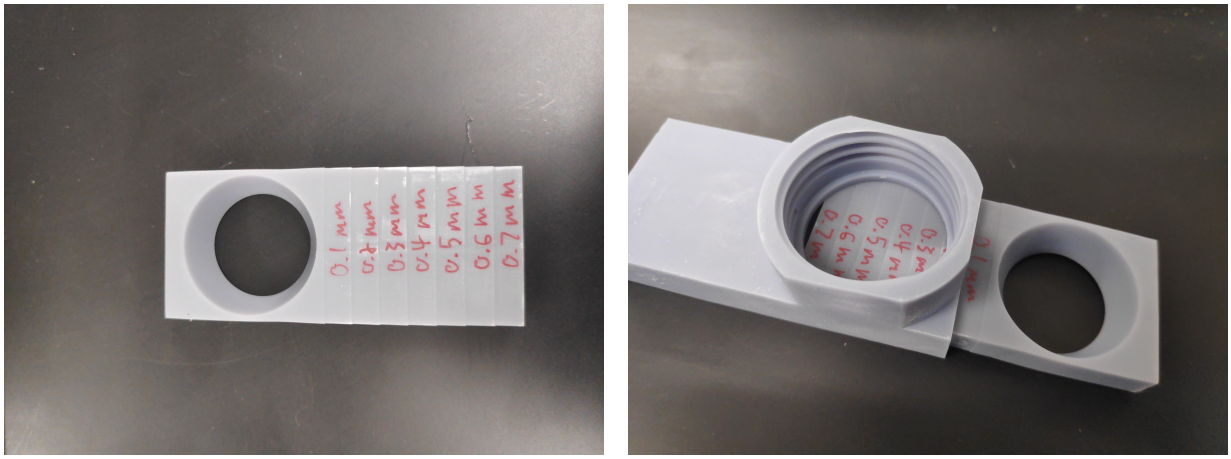
(a) The result of actuating the dispenser one time with SC powder in the source tube showing that approximately 4 mL is dispensed, as expected.



(b) SC leaking out of the back of the powder dispenser.

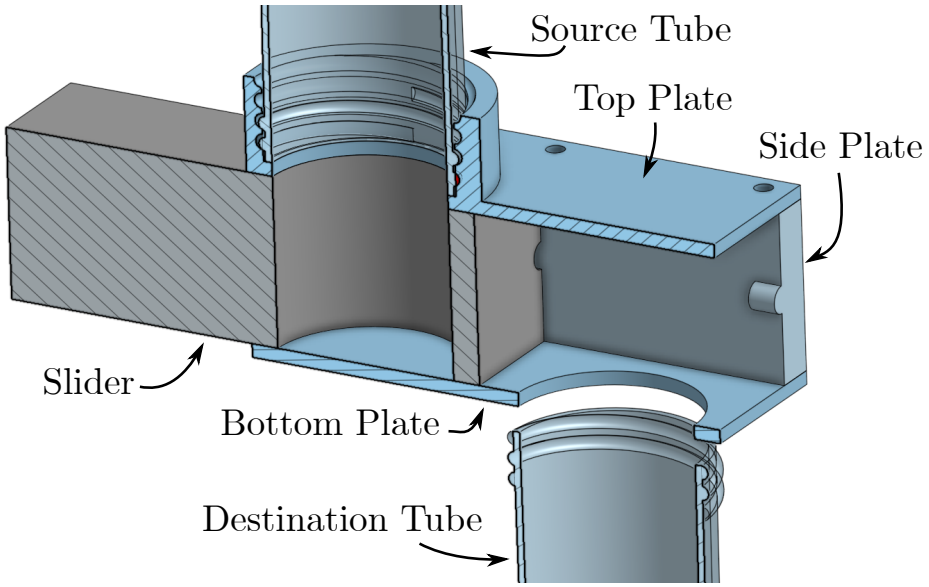
Figure 4.52: Illustration of the results of the experiment testing the refined sliding powder dispenser design using SC powder.

An attempt was made to resolve this by adjusting the clearance between the slider and the housing. In fact, a gauge with multiple clearances was fabricated in order to quickly find a suitable value. This gauge is presented in Figure 4.53. Unfortunately, if the clearance is too small, the fine powder will inevitably enter this gap, jamming the mechanism. In these instances, an incredible amount of force is required to remove the slider from the housing. It was impossible to find a clearance value that would not leak nor jam. Perhaps another manufacturing technology such as machining could produce extremely flat and accurate surfaces that slide smoothly and reliably. However, this is inaccessible and a design that relies on such tolerances is not robust. Thus, this sliding powder dispenser was abandoned.

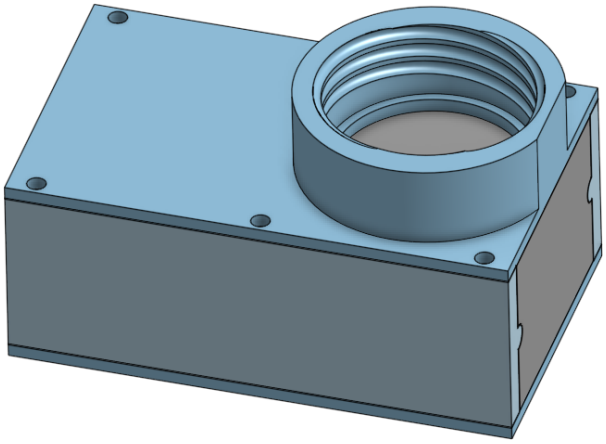


(a) The slider with multiple clearance levels. (b) The gauge slightly inside the enclosure.

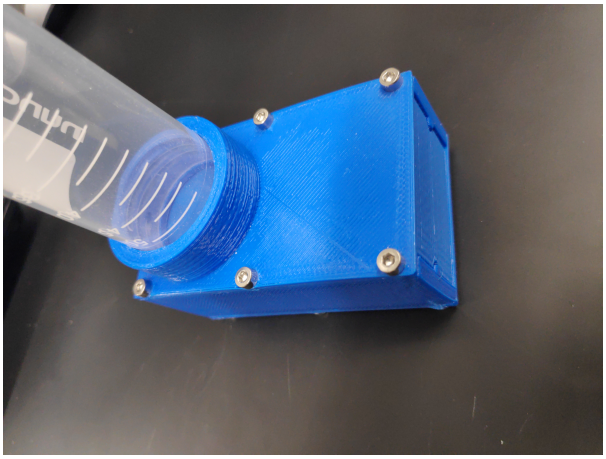
Figure 4.53: Images of the gauge used to quickly find the correct sliding dispenser clearance.



(a) Cross-sectional view showing how the fixed sized cavity transfers powder from the source tube to the destination tube. Annotations define the major components.



(b) Screenshot of the CAD sliding powder dispenser.



(c) Image of the assembled dispenser printed in blue filament with an empty tube in the source position.

Figure 4.48: Images of the prototype sliding powder dispenser.

4.4.2.2 Attempt # 2: Archimedes Screw

After the failure that was the sliding powder dispenser, a more traditional design was attempted. The Archimedes screw is a helical screw design to transfer liquids or powder, or to harvest the energy of a moving liquid. This a well established design was named after Archimedes. There is evidence that variants of this design have been used for millennia [120]. Such screws have since been widely used for powder handling [118], [119]. This design was considered initially, as stated in Section 4.4.2.1, but dismissed as too difficult to manufacture. However, in the interim, the lab has acquired an MSLA resin printer, which should have the required resolution to create such a design. Thus, it was reconsidered and attempted.

The operating principle of the Archimedes screw is that a helical profile pushes the powder from the inverted source tube into the destination tube. This process is illustrated by the cross-sectional view shown in Figure 4.54. A ball bearing is inserted onto the shaft of the screw, which is secured in place by fastening a spur gear. The outer diameter of the bearing has an interference fit into the housing, fully constraining the screw. This gear meshes with its counterpart, which is affixed to the shaft of the stepper motor used to turn the screw. Such a gear pair allows for the motor to be mounted under the dispenser, reducing the footprint, and slightly increase the mechanical advantage.

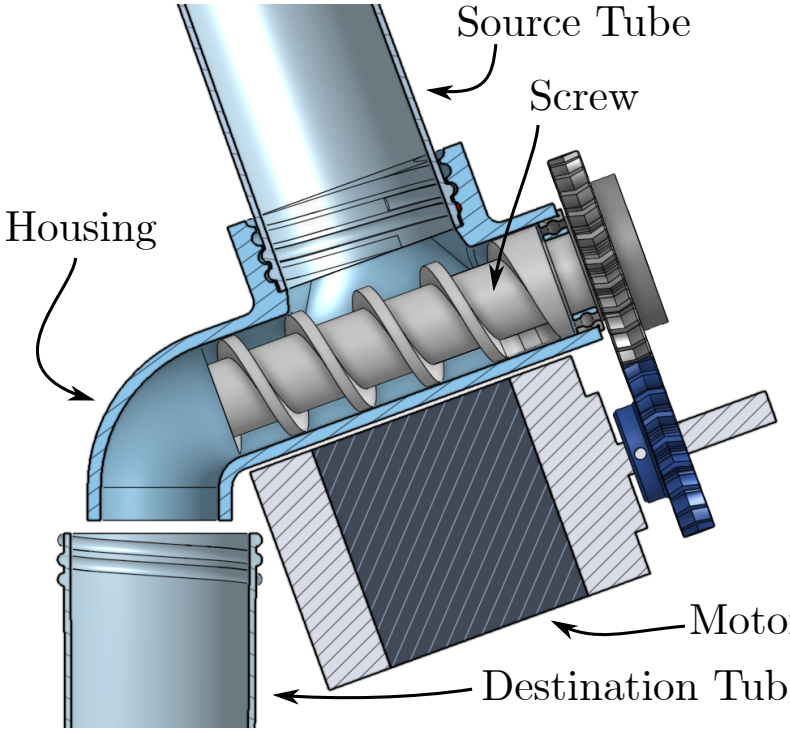


Figure 4.54: Cross-sectional view of the Archimedes screw powder dispenser CAD. The major components are annotated.

The subassembly presented in Figure 4.54 is designed to be inserted into the array shown in Figure 4.55. The dispenser itself can be removed to refill the source tube similarly to the sliding dispenser (Section 4.4.2.1). This is the reason for the gear pair rather than a belt and pulley system; the gears can easily be separated and will mesh anew once reconnected. There is no need to remove and keep track of a belt while the tube is refilled.

The array of Figure 4.55 can easily be adapted for mounting above the *Conveyor* (Section 4.4.3). However, after the failure that was the sliding dispenser, such integrations were not attempted until this design was proven.

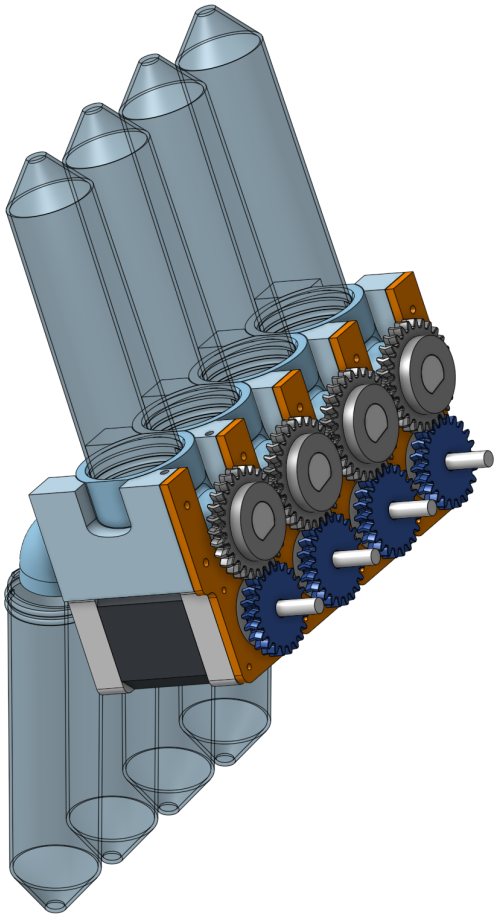
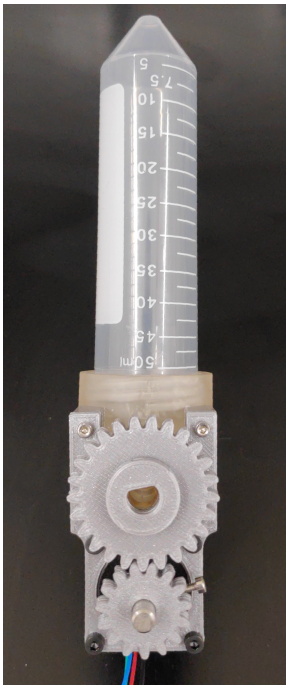


Figure 4.55: Array of four Archimedes screw powder dispensers in their mounts.

The screw and housing of this design was originally intended to be printed using the MSLA resin printer. The assembled dispenser is presented in Figure 4.56. While this dispenser initially appeared to work well, a serious problem was quickly noticed. The resin screw and resin housing were grinding against each other, releasing a fine powder into the dispenser. This is not only a problem because such interference requires a large torque to overcome, but a powder dispenser that wears introducing a contaminant is an obvious deficiency. The cause of this interference is clear: the housing is not perfectly straight. The justification for this claim is that the housing shows wear in exactly three places. If the clearance were simply too small, the wear would be axially symmetrical. If the screw were warped, this component would present three wear marks instead. A visualisation of this issue is presented in Figure 4.57. Due to this problem, alternative manufacturing

strategies were considered that could produce a straight housing in a material that is not prone to wearing. Furthermore, changing the material will enable a more assured chemical compatibility, as the compatibility between the stocks and the photopolymer resin could not be determined due to a lack of public information on the latter.



(a) Front view.



(b) Side view.



(c) The screw, bearing, and spur gear outside of the housing.

Figure 4.56: Photos of the Archimedes screw powder dispenser where the screw and housing are printed using the MSLA printed in photopolymer resin. A previous iteration of the design is shown with a smaller motor and bearing, and narrower central shaft for the screw.

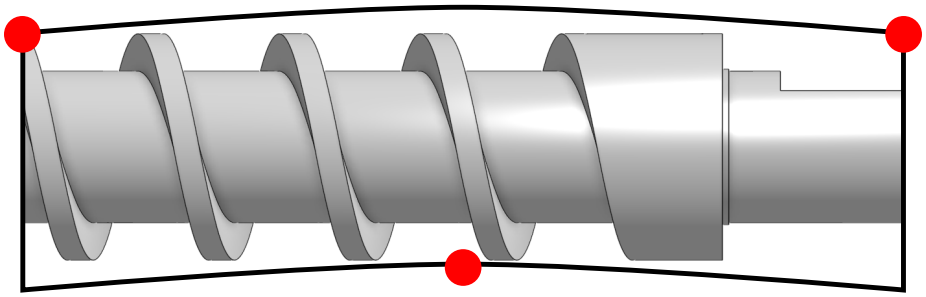
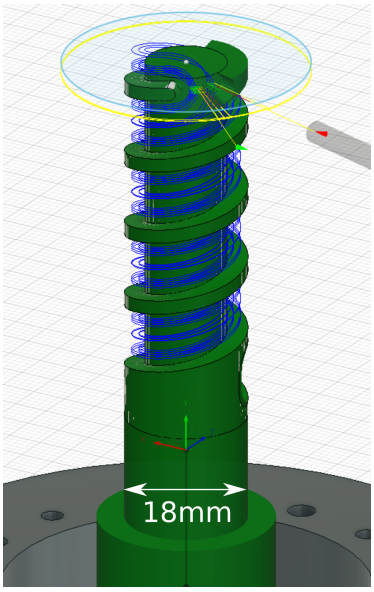


Figure 4.57: Dramatized illustration of the warped Archimedes screw housing, where the wear points are represented by red circles and the warped housing is represented by the black outline.

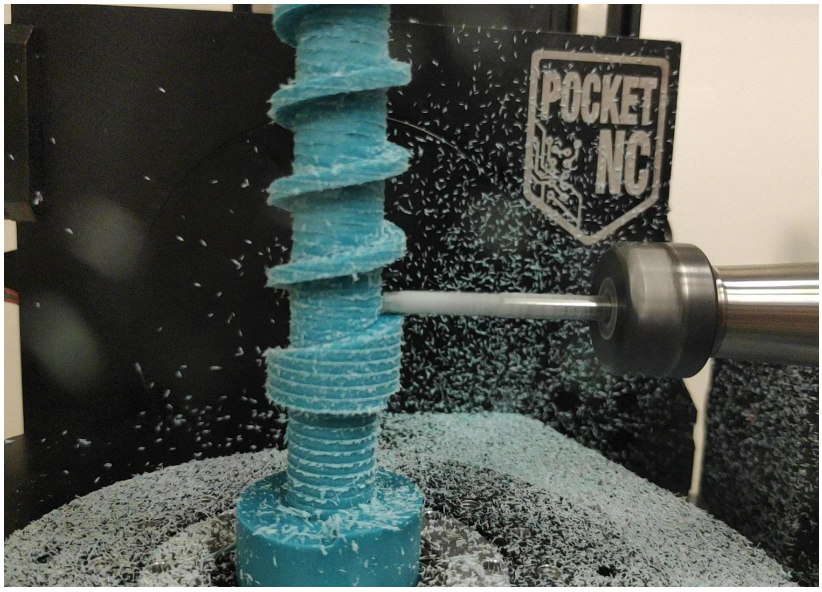
Roberto Romero from the Institute for Quantum Computing (IQC) Tech Lab was consulted about this problem, who proposed two alternative manufacturing techniques. In this lab, it was possible to fabricate screws and housings using FDM 3D printers and using a 5-axis CNC milling machine. The primary reason that the IQC tech lab's 3D printers are able to produce such complex pieces is that these are able to print multiple materials simultaneously. It is thus possible to print soluble support materials, enabling much more intricate designs to be printed with high quality.

Using these alternative manufacturing technologies enables material choices that guarantee chemical compatibility. Fused deposition modeling 3D printing is possible with polypropylene filament [121], the same material from which the 50 mL tubes are made, which are known to be chemically compatible with the stocks required for ATPE. Furthermore, the dispenser's components can be milled from chemically resistant machinable plastic [122].

After receiving training for the CNC mill, it was possible to create two screws in machineable plastic. First, the tool paths for the machine to follow were created by defining a list of machining operations in *Autodesk Fusion 360*. A visualisation of these tool paths is presented in Figure 4.58a. With the tool paths established, the screw can be machined, as presented in Figure 4.58b. Two screws are manufactured to experiment with different machinable plastics and to refine the tool paths for a smoother surface finish.



(a) A screenshot of *Autodesk Fusion 360* showing the tool paths that create the blades of the screw. The outer diameter of the screw is annotated as a reference for scale.



(b) The tool path from (a) being machined inside the *Pocket NC* 5-axis CNC machine.

Figure 4.58: The fabrication of the Archimedes screw using the *Pocket NC* CNC machine.

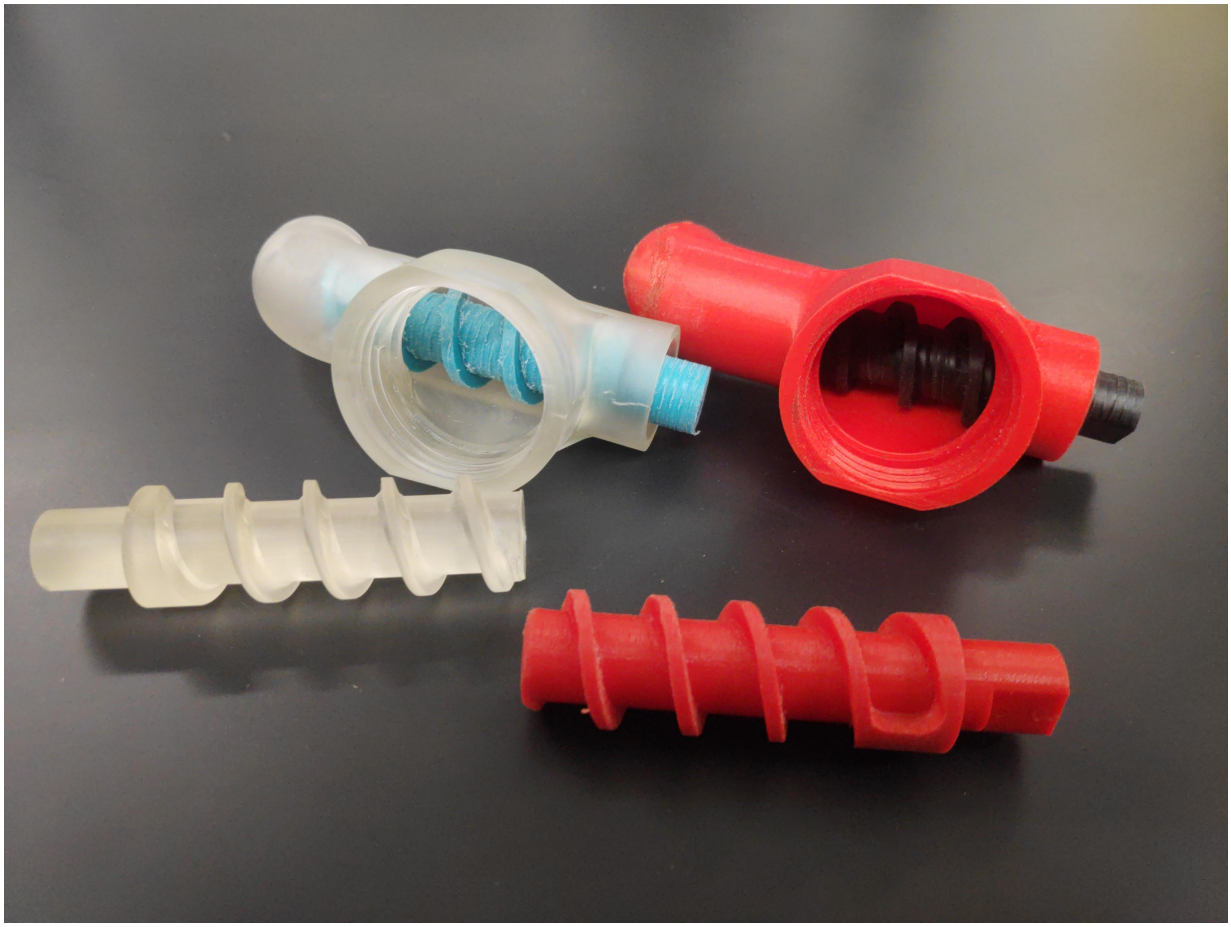


Figure 4.59: A collection of Archimedes screws and housings manufactured using various fabrication technologies. The translucent screw and housing were 3D-printed on a MSLA 3D printers in photopolymer resin. The red screw and housing were 3D-printed on an FDM 3D printers in PLA filament. The blue and black screws were made on a 5-axis CNC mill in machinable plastic.

The initial tests of these new Archimedes screws were very promising. It was possible to dispense PEG and SC without any leaks. With this encouraging result, the repeatability of this dispenser design was tested with the assistance of HeeBong Yang. The motor was programmed to perform a fixed number of rotations and the mass of the dispensed SC powder was accurately weighed. Over the course of this experiment, the motor skipped steps on multiple occasions, necessitating the current limit to be increased. This indicates that the rotary resistance was climbing. The experiment was cut short and the dispenser

was disassembled, revealing that the fine SC powder was clumping together and building up on the inner wall of the enclosure, increasing the resistance of the screw's rotation. An example of this is presented in Figure 4.60. Thus, this design was also abandoned.

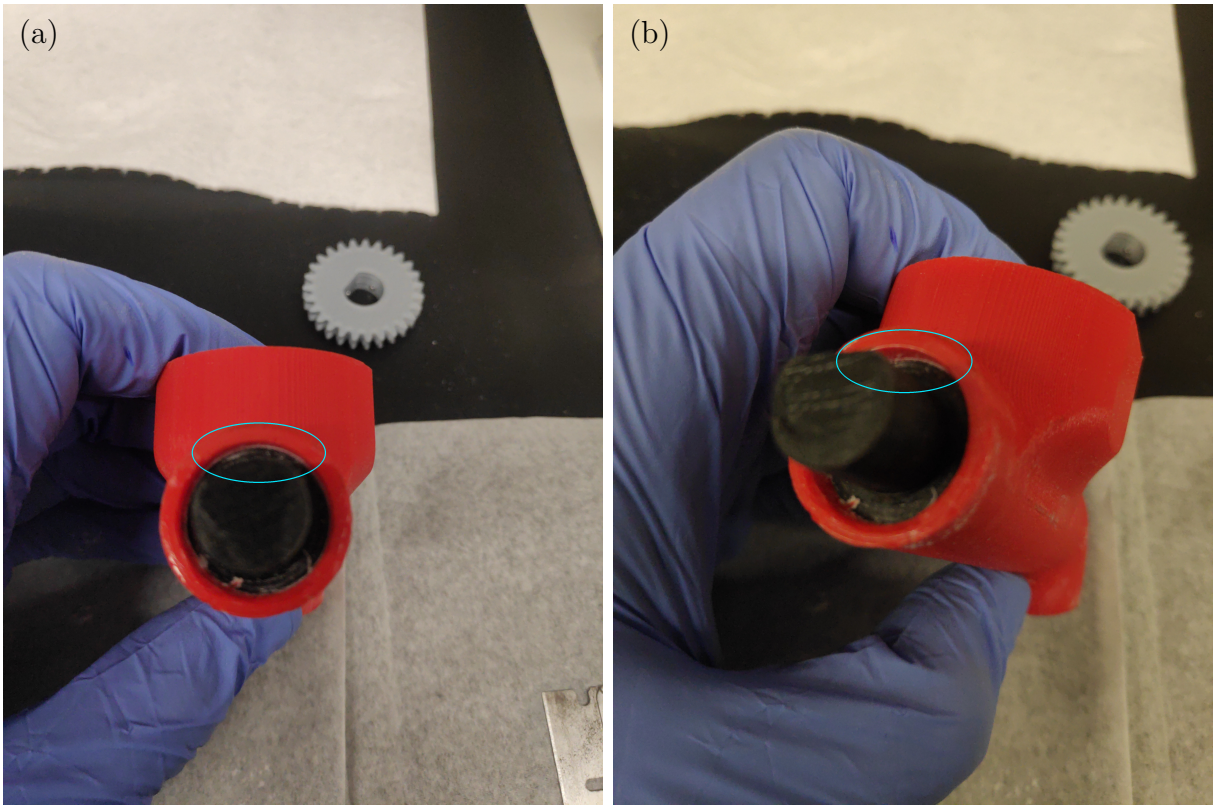


Figure 4.60: Images showing the build up of SC powder while testing the Archimedes screw dispenser. A thin white layer is visible between the black screw and red housing, highlighted by a cyan outline.

4.4.2.3 Attempt # 3: Vibrator and Mesh

The lesson learned from attempts # 1 and # 2 is that any time two sliding surfaces interact with the powder, the latter will inevitably make its way into this gap and jam the system. If the clearance between the sliding surfaces is increased, the powder is not constrained and cannot be properly controlled. Thus, a design was attempted without sliding surfaces.

The third attempted design uses vibration to control the flow of powder through a sieve. The solid cap of the 50 mL tube is replaced with one with a grid of small holes. The

concept is similar to a salt shaker. The tube can be inverted without the flow of particles, which will not be well positioned to enter the cavities. However, when the tube is vibrated, the particles will find their way to holes. The size of the holes can be adjusted for particles of different sizes. An image of this design is presented in Figure 4.61.

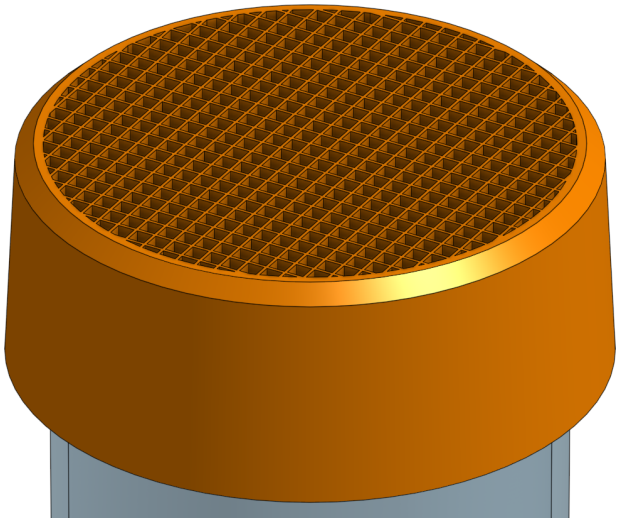


Figure 4.61: CAD screenshot of the designed tube cap with a grid of small 1 mm square holes.

Three prototypes were fabricated with different hole sizes: 1 mm, 0.75 mm, and 0.5 mm. These models were printed on the MSLA resin printer. The largest 1 mm mesh worked well for PEG powder. However, these holes were too big for SC. The powder would flow without vibration, meaning the flow was uncontrollable. Unfortunately, the smaller 0.75 mm and 0.5 mm prototypes did not print well. Liquid resin remained in portions of the mesh, which solidified during the post-print curing phase. This resulted in a mesh that was mostly solid. An image of this issue is presented in Figure 4.62.

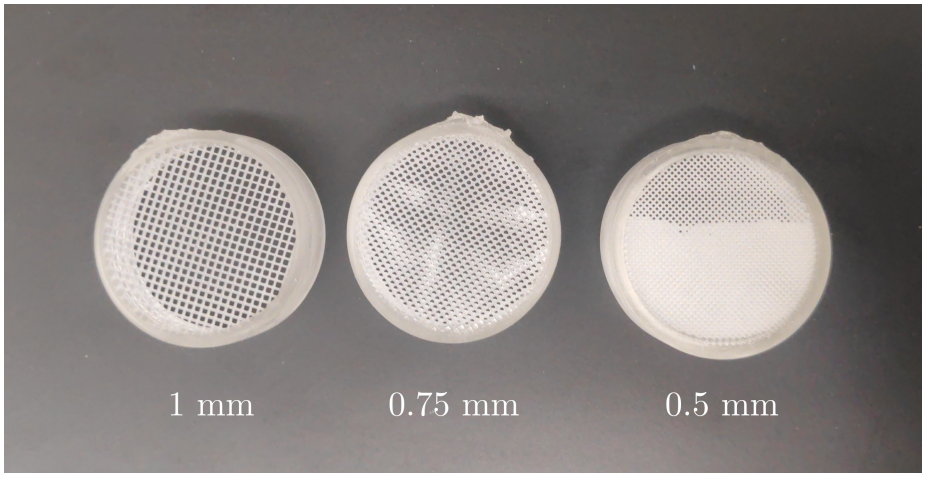


Figure 4.62: Image of the 1 mm, 0.75 mm, and 0.5 mm caps printed in resin. Only approximately the top third of the 0.5 mm cap is free, the bottom two thirds being blocked with resin.

After this problem was noticed, another attempt was made where the part was cleaned even more thoroughly before the post-print curing. Still, it was impossible to remove the resin in the mesh. This resin seemed to be partially cured, which could happen if there is light bleed from the printer's mask. On paper, the resolution of this printer is sufficient to print even finer meshes. Sadly, it seems that such enclosed regions are prone to wicking resin, limiting the detail that can accurately be achieved.

The proposed solution to this problem is to purchase a fine mesh with a similar hole size and to attach it to a cap that is fully open. At the very least, these experiments enabled the discovery of the approximate range of mesh sizes needed for this application. This design is still in the prototype phase. The planned future work includes the addition of a vibratory motor to each source tube, a funnel to carry the powder to the destination tube, and a vibration isolating mount for the source tube. The final element is required so that vibrations of the *OT-2* and adjacent dispensers do not trigger the flow of powder.

4.4.3 *Conveyor*

As stated in Section 4.4.1, there should be a system to bring empty tubes under the various powder dispensers and to store a large number of empty and filled tubes. Of course, many alternatives exist for this task. An option that solves both constraints simultaneously and

that can be implemented simply is a conveyor. The strategy is to create a closed chain where every link can support a tube. This chain can be rotated through a series of cogs to move any tube to any position along the chain. By positioning the dispensers between two cogs, a tube can be moved under any dispenser and filled via gravity. Such a conveyor was designed and its 3D model is presented in Figure 4.63.

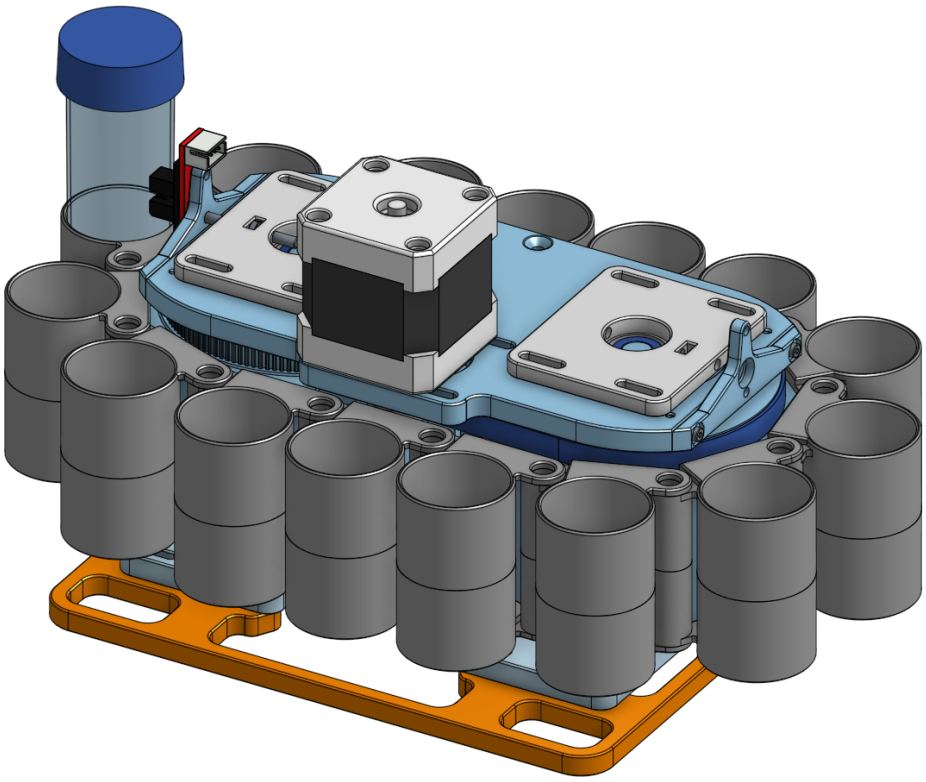


Figure 4.63: A screenshot of the CAD model of the *Conveyor* used in the *Powder Handling System*.

The *Conveyor* is made of 15 3D-printed links, each of which having a hole for a tube. The holes are tapered at the bottom, matching the tubes, so that the latter do not fall all the way through, as presented in Figure 4.64. The links interlock with each other, each having two tabs that span the last until a closed loop is formed. The connections are achieved through the use of plastic threaded barrels and screws [123]. This provides a smooth surface, creating a hinge that allows rotation between links.

The links of the *Conveyor* mesh with two cogs, one on each extremity. The cogs have cylindrical grooves that interface with the columns of the links. Furthermore, the links

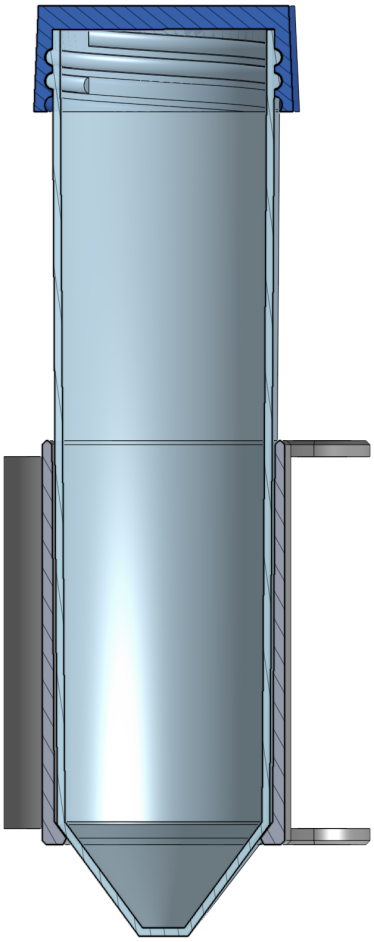


Figure 4.64: Cross-sectional view of the link and tube, featuring a taper at the bottom.

have a cutout below the top surface that rides along on the cogs, constraining the links vertically. At the top and bottom of each cog is a small shaft that is inserted into the centre of a ball bearing. The bottom shaft is designed as a separate piece that is press-fit into place. This way, the cogs can be 3D-printed without support material. Finally, the cogs possess embedded teeth for a timing belt attached to a drive motor. Although only one cog is driven (the other turns via the chain), both cogs include the timing belt teeth. They do not interfere and this reduces the number of unique parts. An image of the cog is presented in Figure 4.65.

Each cog is supported by a bracket on the top and bottom. These brackets are mounted to the main top and bottom plates via slots, which allow the distance between the cogs to

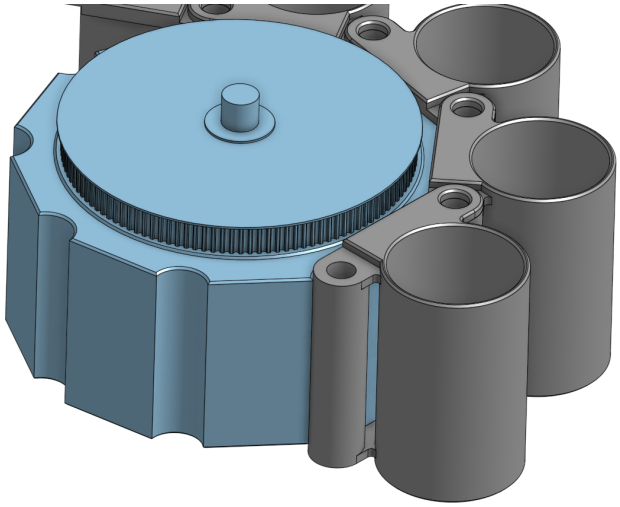


Figure 4.65: A screenshot of the CAD model of the cog with multiple links. The remaining links are hidden to show the teeth of the cog.

be adjusted in order to tune the tension of the *Conveyor*. To facilitate this, the horizontal position of each bracket is defined by a long bolt attached to a small plate at the end of the *Conveyor*. Tightening this bolt will pull the cogs away from each other. This is advantageous as it does not allow the bracket to move freely, losing the original position, when the bolts in the four slots are loosened and because greater mechanical advantage can be achieved compared to pulling the cogs apart by hand while simultaneously attempting to tighten the four bolts to lock the position into place.

The main top and bottom plates are designed to be 3D-printed. The bottom plate has groves to interface with the deck of the *OT-2*, locking it into position as shown in Figure 4.66. The two main plates are connected by columns with embedded nuts that hold the system together.

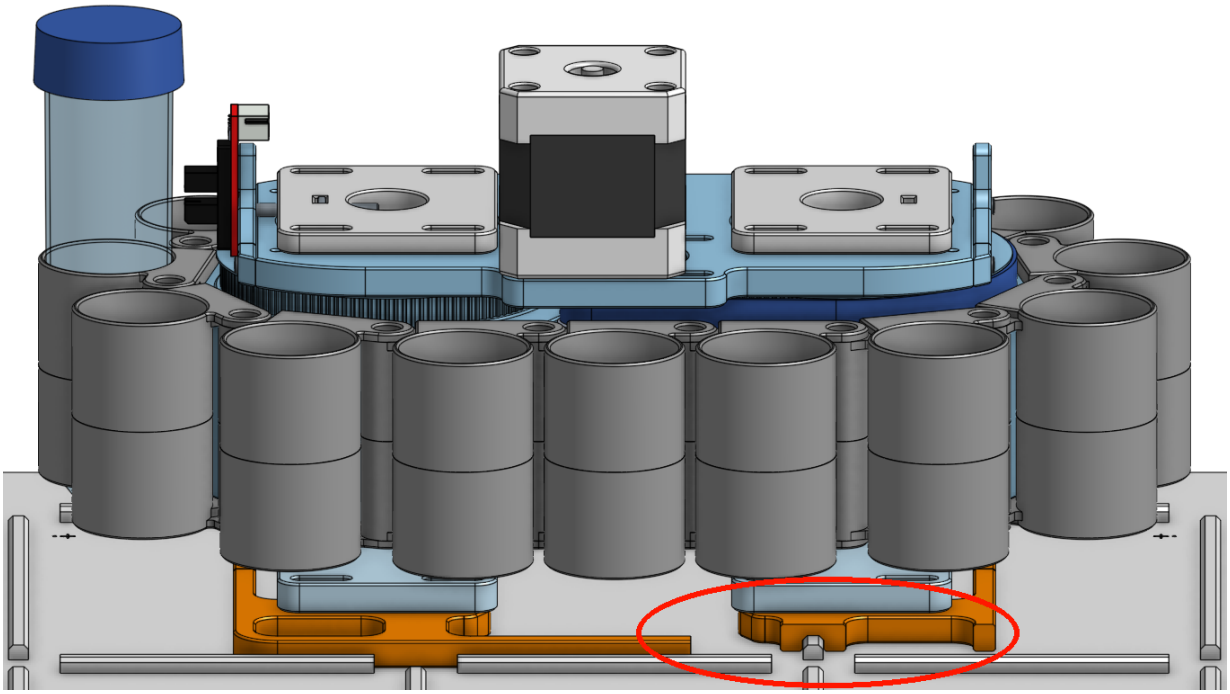


Figure 4.66: A CAD screenshot showing how the orange bottom plate interfaces with the deck of the *OT-2*. Part of this plate is cut away (red outline) showing the grove added into interface with the ridges of the deck (Figure 4.2).

The motor used is the ubiquitous NEMA 17 [103]–[105] stepper motor identical to the one used in the *Vial Opener* (Section 4.2). A 16-tooth pulley is attached to the shaft, which in turn meshes with the timing belt and 120 teeth of the cog. This tooth ratio increases mechanical advantage and reduces speed by a factor of 7.5. The motor is mounted to the main top plate of the *Conveyor* using 4 bolts in slotted holes, allowing the tension of the belt to be adjusted.

Finally, an optical end stop is used to detect the zeroth link. This is a small off-the-shelf device consisting of a light source and a light detector separated by a small cap. The zeroth link is special and possesses a small arm that passes through this beam of light, which is recognized by the detector. It is important to accurately determine the centre position of the zeroth link because stepper motors lose their position when power is lost. The index of the links must be determined to know which substance is in which position, and the centre of the link must be determined to accurately position the cups for the dispensers and *Gripper*.

Once the design was complete, the components were ordered and the 3D models were printed. At this point, the *Conveyor* was assembled and the motion was tested. The finalised *Conveyor* is photographed in Figure 4.67. However, the *Conveyor* has not yet been integrated with a dispenser system as no viable design for the latter yet exists, as discussed in Section 4.4.2.

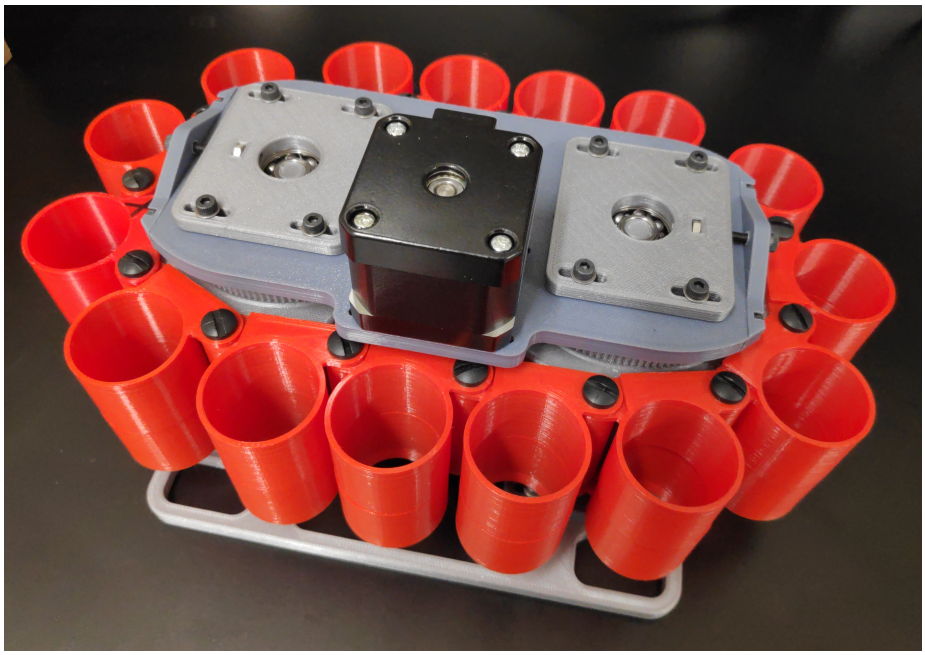


Figure 4.67: Image of the fully-assembled conveyor

4.4.4 *Weigh Module*

The balance selected is the *Mettler Toledo WKC205C*, which was selected for its small size, high accuracy and capacity, and programmable interface. This *Weigh Module* was designed to be integrated into a machine, not used manually. Thus, there is no display, pan, or draft shield. Instead, the weight is read over the RS232 or RS422 industrial communications protocols [124]. Furthermore, the selected balance has a maximum capacity of 200 g and a 0.01 mg readability [124]. This meets our requirements for ATPE as established by HeeBong Yang of 60 g capacity and 0.01 mg repeatability. All of these factors make this model perfect for embedding in the *OT-2*.

In order to integrate this balance into the *OT-2*, two non-destructive alterations were

made. First, a small stage was 3D-printed with groves on the bottom to lock into the *OT-2*'s deck (the *OT-2* positioning system is presented in Figure 4.2) and with columns protruding from the top that mate with the holes in the four corners of the *Weigh Module*. Second, an adapter was 3D-printed making the weighing pan of the balance compatible with the 50 mL tube. Only the powder for the stocks need to be weighed, so it is not required to support the vials. An image of the balance with these two augmentations is presented in Figure 4.68.

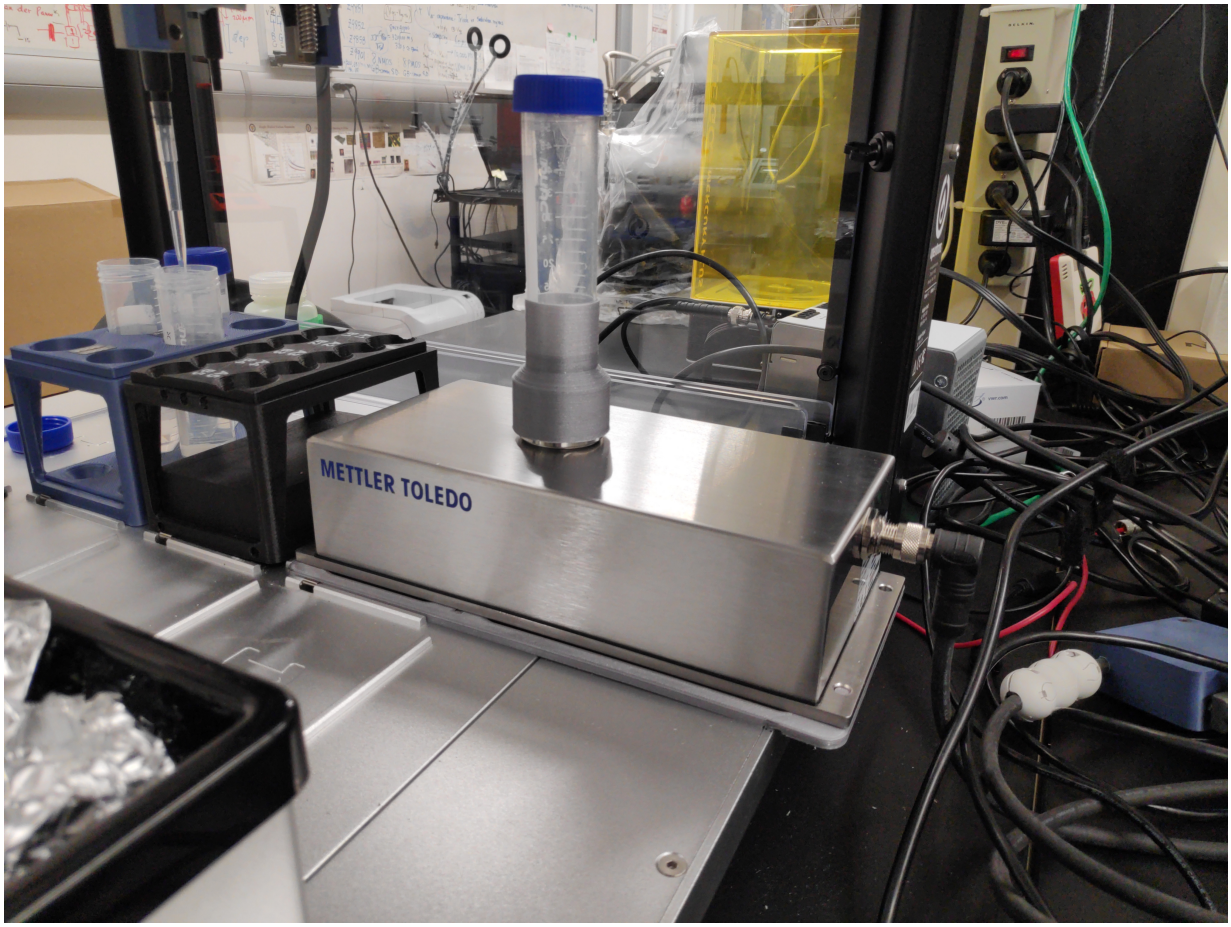


Figure 4.68: Image of the balance in the *OT-2* using the stage to interface with the *OT-2* deck and adapter for a tube.

Additionally, to connect the balance electrically, the un-terminated cable provided had to be wired to the *OT-2*. A USB to RS232 cable (a cable with USB A on one end and RS232 with the standard DB9 physical interface on the other with a translation chip in

between) and a DB9 to screw terminal break out were used. The latter was integrated into a small 3D-printed electronics box, inside which the wires of the *Weigh Module*'s cable were wired into the screw terminal. This way, the USB adapter cable can be connected to the box in order to interface with the *Weigh Module*.

The balance was calibrated and tested with the assistance of Prof. Na Young Kim. The lab does not possess a set of precision known masses, so the balance was calibrated to samples weighed in the *Mettler Toledo* balance used for manual ATPE. The reasoning here is that the manual balance is calibrated sufficiently well for ATPE (or the recipe accounts for its error), and that the two match very well once calibrated, so the *Weigh Module* must also be calibrated well enough for ATPE.

After the calibration, reading the weight programmatically was tested. For this, the Python package `mettler_toledo_device` [125] was used, which is a library for interfacing with *Mettler Toledo* devices over RS232. This system was used to make stocks, although since a viable dispenser is still lacking (Section 4.4.2), the approximate amount of powder was added manually. The inserted powder was weighed by the *WKC205C* and the correct amount of deionized water to reach the target concentration was added by the *OT-2*. This experiment shows conclusively that there was no difference in the results when the stocks made semi-autonomously were used in ATPE, validating the *WKC205C* for this application. The result of this second step is presented in Figure 4.38.

4.5 *Inventory System*

The *Inventory System* has the goal of tracking the contents and history of each sample and stock through the use of unique identifiers in the form of barcodes. This component is responsible for the printing and scanning of barcodes, as well as the code and database responsible for tracking the contents and history of each vial via its barcode.

For ATPE process samples, the barcode must reference the current contents of the vial, the history of all ATPE steps performed on it, the parent sample, the type of CNT used, and other information required to identify vials. Furthermore, each tube holding a stock must have a code in a different format referencing the contents of the tube and the date that the stock was mixed.

Having an identifier on each vial is crucial for traceability; after a vial is removed from the *OT-2*, it can easily be mistaken for another if it is not marked. Additionally, while hybrid ATPE is still being performed (Section 6.1) or during a manual intervention of the future fully-automated system, it is important to identify the vials. For instance, during

hybrid ATPE, the vials are manually removed from the *OT-2* and placed in the centrifuge. If these are properly marked, they can be re-inserted into the *OT-2* in any order. The first step after the protocol is resumed will be to scan and identify each vial.

This bar-coding system has many components that must interact. First, an identifier should be generated for each container, which is the responsibility of the *Central Command* software. This identifier must be printed in the form of an image, which is possible through the use of a computer-connected label maker, and applied to the container. A *Barcode Scanner* must be included inside the *OT-2* so that these codes can be recognized. Finally, the ATPE recipe software must be made aware of these identifiers.

As will be seen, the *Barcode Scanner* and *Barcode Printer* are currently not finalised. However, ATPE can still be performed. The samples can be marked with a pen, which is how tracking was done during manual ATPE before this automation project. Furthermore, the samples can be tracked in *OT-2* protocols by their location.

4.5.1 Identifier Format and Generation

The format of the identifiers to be printed on the barcodes is `SG65-20220307-001-00` for process vials and `SDS-20220307-1305` for stocks. The meaning of each block is given in Table 4.7 and Table 4.8, respectively. This format was proposed by HeeBong Yang based on his vast ATPE experience. The justification to have such data in the code rather than a simple incrementing number, which would function identically for the purpose of reference information in the database, is that this format has meaning without the need to look anything up. The most important pieces of information are directly visible in the identifier, which is printed in plain text under the computer-readable code.

Table 4.7: Example identifier for a process vial and meaning of each block. Specified by HeeBong Yang.

SG65-20220307-001-00	
SG65	SWCNT powder
20220307	ATPE start date in the format `yyyymmdd`
001	Order of ATPE in a day
00	Subsidiary vials

Table 4.8: Example identifier for a stock and meaning of each block. Specified by HeeBong Yang.

SDS-20220307-1305	
SDS	Polymer, surfactant, or mimic identifier
20220307	Creation date in the format `yyyymmdd`
1305	Creation time in the format `hhmm`

QR codes are generated programmatically using the `qrcode` Python package [126] during automated ATPE. Additionally, it is often convenient to quickly generate a QR code for debugging or validation purposes. In this case, it is possible to use the `qrencode` command line utility [127], which is available for many Linux distributions.

4.5.2 *Barcode Scanner*

Multiple options were considered for the *Barcode Scanner*. A retail-style handheld 1-dimensional bar code scanner could be used, though this may not be reliable on the small curved surface of the vial. Several industrial barcode scanners are available, which boast many advanced features and support for 1D and 2D codes. The companies *Keyence* [128], *Cognex* [129], and *Banner Engineering* [130] were contacted for quotes on their industrial barcode scanners. However, the pricing was prohibitive. Finally, it would be possible to used a *Raspberry Pi Camera Module* [131] or a webcam and recognize the codes through *OpenCV* [132] or a similar Python library. This solution uses components that the lab already possesses, so is the most affordable, and offers the most flexibility. In the event that this method is insufficient, it will be an affordable and easy prototype to determine the system requirements and limitations. Thus, this option was selected.

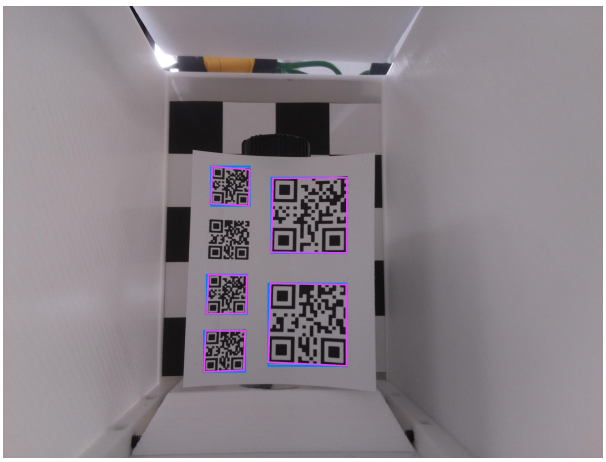
4.5.2.1 Preliminary Testing

The initial tests of a custom barcode scanning solution used 1-dimensional codes like those commonly used in retail. The Python library *OpenCV* [132] was used to locate and recognize the identifier from an image of the barcode. Specifically, the `barcode_BarcodeDetector` module of `opencv_contrib` [133] was used with code inspired from Liang and Wang [134]. This method worked well for simple cases, but was not reliable if there was any distortion in the image.

Therefore, 2-dimensional quick response codes (QR codes) were tested, the reasoning being that this kind of code has markers to identify the edges that make it more robust to distortions. Although *OpenCV* has the ability to detect and decode QR codes [135], [136], it was found to be unreliable and sensitive to distortions. Therefore, the *pyzbar* Python module [137], which uses the *ZBar* library [138], was used. This module has proven to be much more robust. An example of its output on exemplary codes is presented in Figure 4.69, where it is possible to see that QR codes are recognized even if they are at an angle or if many small codes are present. In Figure 4.69b, it is possible to see that one small code was missed. However, after cropping the image so that no other codes are visible, it was correctly recognized. Therefore, this is not a problem for this application, since only one code will prominently be displayed in the image and since the image can be cropped due to the repeatable position of the code in the image.



(a) A large QR code photographed inside the *Photo Stage* adhered to a vial on an extreme angle to test the robustness of the code recognition.



(b) Multiple small QR codes photographed inside the *Photo Stage*. The smallest codes on the left have an appropriate size for the 50 mL vials.

Figure 4.69: Results of preliminary tests of 2D QR codes. The blue and pink outlines are added by the image processing software and represent the aligned bounding box and the transformed outline of the code, respectively. The encoded data was also correctly extracted for all detected codes.

4.5.2.2 Integration

The plan for the integration of the barcode scanner into the *OT-2* is to mount a high-resolution camera above the liquid handling robot such that the barcode of every container in the machine can be read simultaneously. The location of the code in the image can be used to determine the position of the container, including the deck slot of the container's rack and the column and row of the container in the rack. The top panel of the *OT-2* is made of transparent acrylic, so a camera mounted above this panel would capture all deck positions.

An alternative option of using a smaller, lower resolution camera mounted to the gantry of the *OT-2* was also considered. However, this option was ultimately rejected for its two major drawbacks. First, the camera would not be able to capture an image of every deck slot simultaneously. The plan would be to take multiple pictures, moving the gantry between them. This adds additional time compared to capturing all samples in a single image. Second, it would be challenging to find space on the gantry for the camera where it would not collide with the side panels of the *OT-2* and it would be challenging to connect additional cables to the gantry. Both of these drawbacks are solved by using a high-resolution camera above the *OT-2*.

Multiple camera options have been tested, but unfortunately, none have had the required resolution to detect a QR code the size of a 14.7 mL vial at this distance. Images from a webcam with a resolution of 1920×1080, a cell phone with a resolution of 4032×3024, and a *Raspberry Pi*-compatible camera [139] with a resolution of 2592×1944 were all tested, but none of the codes from any of these images was properly detected. The *Raspberry Pi*-compatible camera supports a C- or CS-mount lens, so the stock lens was replace with another having a fixed focal length of 12 mm, which was purchased for a different project. This allowed the codes to be detected, but with this focal length, the image was zoomed in such that only one slot of the deck was visible.

The recommendation of the author is to use the *ArduCam 64MP Raspberry Pi*-compatible camera [140]. This camera has nearly four times as many pixels as the highest resolution cameras tested thus far. It is affordable, and has the same form-factor as the official *Raspberry Pi Camera Module 2* [131]. Alternatively, it may be possible to use the official *Raspberry Pi High Quality Camera* [141] with an appropriate lens. This camera only has a resolution of 12.3 mega-pixels, which is similar to the cell phone used previously. However, if an appropriate lens is used to frame the image properly, this resolution may be sufficient. This was validated by bringing the cell phone camera closer to deck of the *OT-2* such that the edges of the image were aligned with the edges of the *OT-2* deck. At this distance, all codes were properly recognized, except for one code in only image. Using *ArduCam*'s online

focal length calculator [142], it was determined that the lens should have a focal length of 7.6 mm for this camera given the width of the *OT-2* deck and the distance between the top of the vial where the code with be adhered and the top piece of acrylic.

4.5.3 *Barcode Printer*

The printer used to realise the barcodes is the *Brother QL-710W* [143]. This model is used because the lab already possessed it and because it can be used programmatically over USB. All of the QR codes presented were printed via the Python package `brother_ql` [144]. In this way, it is possible to print identifiers directly from the *Central Command* software.

4.5.4 Tracking Code

Perhaps more important than the *Barcode Scanner* and *Barcode Printer* is the code and database used to store all the information about a sample. While the former are useless without this software component, the tracking code is already being used to keep a history of vial contents in recent hybrid ATPE attempts (Section 6.1). The only difference between the current implementation and the intended behaviour is that presently, each sample must be registered by location and barcode, whereas in the future, it should be possible to specify only which barcodes are used for a protocol, and the *Barcode Scanner* will identify the locations of these.

The liquid tracking code must store information about the contents and history of each sample in a database and must also recognize when aspiration and dispense events occur in order to keep this information up to date. The database chosen is SQLite [145]. Its simplicity and small footprint makes it a good choice for embedded applications such as this. The schema of the table used to track vials is presented in Table 4.9, which shows many similarities to the barcode identifier format of Table 4.7. A similar, simpler table must be established to track stocks. When deciding between tracking polymers and surfactants by volume or by concentration, a trade-off had to be made. If tracking is done by volume, the result of dispensing the solution into a vial is easy. Simply add the volume of each chemical in the vial with the volume of the same chemical in the pipette. However, aspirating from a vial is difficult. The volume of each chemical must be converted to a concentration based on the total volume in the vial, and then back into a volume based on the total volume aspirated. The volume of each chemical remaining in the vial must be updated in a similar manor. Finally, computing the volume of each chemical in each phase during separation is easy if it is assumed that the top phase has 100 % of the vial's PEG

volume and that the bottom phase has 0 %. The same logic follows for the other chemicals depending on whether they are partitioned to the top or bottom phase; PEG is used as an example. Symmetrically, if tracking is done by concentration, aspirating is easy, dispensing is challenging, and separation is challenging. Thus, the chemicals are tracked by volume. Also note that the chemical volumes are of the diluted solution, not the raw chemical. This simplifies the calculations and makes the volume easier to track based on how much of a diluted stock was aspirated. Ultimately, the value of interest is the concentration of the raw chemical, but this is easy to calculate.

Table 4.9: Format of the SQLite table tracking all information for ATPE main process sample vials.

Column	Description
`id`	A number identifying the row.
`barcode`	The identifier embedded in the QR code.
`parent_barcode`	The identifier of the parent sample.
`timestamp`	The date and time this change to the sample's contents was changed.
`step_identifier`	The identifier of the step (T, 2B, etc.) in the format developed by Fagan described in [68].
`cnt_type`	The type of CNT used initially, like SG65i [75].
`sds_volume`	The volume of SDS in the sample.
`doc_volume`	The volume of DOC in the sample.
`sc_volume`	The volume of SC in the sample.
`peg_volume`	The volume of PEG in the sample.
`dx_volume`	The volume of DX in the sample.
`total_volume`	The total liquid volume of the sample.
`step_number`	The number of ATPE steps performed.
`subsidiary_vials`	The vial's number for this execution of ATPE

A challenge of this tracking code is that aspiration and dispense events should automatically be detected. Were it obligatory as the protocol programmer to remember to update the vial contents, it would be easy to make miscalculations or to forget to update the database altogether. Thankfully, the *Opentrons* software exposes a publish/subscribe messaging system [146] where dispense and aspiration events are published, which include the location of the event and the volume transferred. Since the tracking code knows the location of all samples, listening for these events is sufficient to perfectly track the contents

of every vial at all times. The vial must only be registered once at the beginning of the protocol, then the programmer need not remember to update the vial contents.

The code also has many edge cases to contend with. For example, air gaps (often, a small amount of air is aspirated after aspirating liquid to act as a buffer, preventing drips), must be detected and handled. Air gaps trigger normal aspiration events in the *Opentrons* publish/subscribe system, so air gap events are recognized by the pipette tip's being above the brim of the vial. Once detected, these are handled by treating the pipette's contents as a stack. When aspirating, the fluids is pushed onto the stack. When dispensing, fluids are popped off of the stack until the volume being dispensed is reached. In the case of air gaps, the fluid "air" is pushed onto the stack rather than the contents of the vial.

One circumstance that cannot be handled automatically is separation. It is not possible to know if a given aspiration event is from the top phase, the bottom phase, or a mono-phase system. Once a vial undergoes centrifugation and two phases are formed, it is photographed in the *Photo Stage* and the boundary between the two phases is detected automatically with computer vision (Section 4.3). The height of this boundary can be given to the tracking code in order to enter the separation mode. In this mode, the vial is considered as having two different solutions. The top phase has all of the vial's PEG and SDS while the bottom phase has all of the vial's DX, DOC, and SC. Aspirations above the detected boundary height will be taken from the top phase solution and aspirations below will be taken from the bottom phase solution. This is implemented using a Python context manager with the code `with sample.separate(height):`, where `sample` is the object returned when registering a sample, meaning that the mode will automatically be reset when the `with` block is left.

4.6 Software

This section discusses the various software required for the ATPE automation. First, the firmware, related libraries, and communications protocol used between all custom modules and the main computer are discussed in Section 4.6.1. Next, the libraries developed for use with the *Opentrons* protocols are detailed in Section 4.6.2. Finally, the *Manual Control Interface*, a GUI put in place to facilitate the debugging of protocols and the development of custom modules, is shown in Section 4.6.3.

4.6.1 Firmware and Communications

The firmware for all custom modules uses the Mbed [102] hardware abstraction layer and is written in C++. The PlatformIO framework [147] is used to facilitate the compilation, library management, and code organisation which not imposing a particular integrated development environment. The STLINK integrated circuit of a Nucleo development board is used to flash and debug the firmware [148].

The custom modules all use very similar firmware. By standardising the microcontroller, USB to UART converter, and other components, it is possible to reuse a large amount of code between modules. The shared code mainly takes the form of reusable libraries. Each library is described in the following sections.

4.6.1.1 Communications Library

The communications protocol and libraries (C++ and Python) are standardised between all custom modules. "Commands" are messages sent from the *Central Command* module to a microcontroller and "responses" are messages sent from the microcontroller to the *Central Command* module in response. The microcontrollers may only respond to messages initiated by the *Central Command* module. In other words, there should be no response message sent without an accompanying command message.

Before implementing the communications protocol, the decision between a binary or ASCII (text) format had to be made. The advantage of the ASCII format is that it is human-readable and it that is possible to manually construct commands and to send them using any UART terminal, which can be invaluable for troubleshooting. However, the benefits to a binary format are numerous. First of all, no special encoding or decoding is required to transmit numbers. It is not necessary to convert numbers to their decimal representation and back to binary again. This reduces complexity and is much more efficient computationally and in terms of message length. Second, it is possible to included a checksum to verify that messages are transmitted correctly. The advantage of the ASCII format is that it is easy to construct a message manually, and manually calculating a checksum would not be very convenient. Finally, it is possible to define a strict and efficient format. It is not necessary to inefficiently separate data by spaces, nor to implement a parser that is robust to extra spaces or various types of white-space. Thus, the binary format was selected. The advantage of the ASCII format — that commands can manually be constructed — is regained when the Python abstraction layer is in place, which is the case for all custom modules. In this way, commands are manually constructed by executing

Python functions, which is superior to an ASCII protocol in many ways. A Python read-eval-print loop (REPL) (the interactive command-line interface accessible by running the `python` command with no arguments) can be used to interactively send commands to a module. This has all the advantages of the binary format, and has some advantages over the ASCII interactive terminal as well. The parser is implemented by Python, which is easier and more reliable than implementing a custom parser, and the syntax is familiar. Furthermore, the REPL is more powerful than a simple UART terminal. It is possible to use advanced structures like loops at the same time as manually sending commands. Finally, the same code used in the interactive REPL can be copied directly into the final code. There is no need to learn the syntax of the Python abstraction as well as the syntax of the communications protocol. This powerful troubleshooting and development experience is offered through the manual control interface discussed in Section 4.6.3.1.

Commands and responses have very similar formats as can be seen by Table 4.10 and Table 4.11. Every message starts with its own length to inform the recipient of how many bytes to allocate and read. This has the consequence that messages cannot be longer than 255 bytes (the maximum value of an unsigned byte), but this has never been a problem. Furthermore, every message ends with a 4-byte cyclic redundancy check (CRC32). This is a system to verify whether a message has been received correctly before executing it.

The second byte of every command is the "command code", a unique identifier of the command to execute. There is a set of base commands that every module must implement, and each module has its own set of specific commands. The base commands are given in Table 4.13. Symmetrically, the second byte of every response is the "response code" indicating whether the command was executed successfully. The response code is an enumeration type [101] whose variants are defined in Table 4.12.

Finally, each message may contain a block of data of variable length. In the case of commands, this block contains the arguments of the command. For example, if the command is an instruction to move to a specific location, the list of arguments would include only the location. In the case of responses, this block contains the response data. For example, if the command is an instruction to read the value from a sensor, the response data would be the sensor reading.

Table 4.10: Format of a command message. The data are sent from left to right as they appear in the table.

Message Length	**Command Code**	**Arguments**	**Checksum**
1 byte	1 byte	Dynamic	4 bytes

Table 4.11: Format of a response message. The data are sent from left to right as they appear in the table.

Message Length	Response Code	Response Data	Checksum
1 byte	1 byte	Dynamic	4 bytes

Table 4.12: Variants of the "response code" enumeration.

Variant	Value	Description
`ACKNOWLEDGE`	0	The command was understood and no errors were encountered while processing it.
`INVALID_CHECKSUM`	1	The checksum of the command did not match the checksum included in the command.
`INVALID_COMMAND`	2	The command code did not match any command code known by the firmware.
`ERROR`	3	A module-specific error occurred. The response data will have a 1-byte module-specific error code.

The base commands must be supported by all modules and are implemented by the communications library. The list of base commands is given in Table 4.13. The commands in the range `0x2X` are relevant to the communications library itself, which is why they are base commands. The commands in the range `0x3X` are general microcontroller commands. These can be dangerous and should only be executed manually during troubleshooting or reprogramming.

Table 4.13: Command names, codes, and functions of the base commands.

Command Name	Code	Function
`WHOAMI`	`0x20`	A request for the microcontroller to identify itself. When the microcontroller is connected, the *Central Command* module does not know which module it belongs to, so the microcontroller must introduce itself.
`GET_STATUS`	`0x21`	Retrieve the current status of the microcontroller.
`RESET_STATUS`	`0x22`	Reset the status of the microcontroller to the default. This should only be used manually to clear an error after the problem leading to the error has been resolved.
`RESET_CORE`	`0x30`	Reset the microcontroller. This is similar to pressing the reset button. This command is only used manually for troubleshooting.
`JUMP_TO_BOOTLOADER`	`0x31`	Jump away from the custom firmware into the microcontroller's bootloader. When the bootloader is being executed, it is possible to flash new firmware without an external programmer. However, this is not as reliable as using STLINK.

Many of these commands relate to the microcontroller status, another notion of the communications library. When using the library, it is obligatory to specify a default status and to update the `current_status` attribute when the status changes. Each module will have a different status enumeration type [101] with an "OK" default status and appropriate error statuses.

The reason to implement this status system and not simply to use the response data to send the status is that many commands operate asynchronously. The command sets a slow action in motion that will not fail immediately but may fail before its completion. A synchronous command works well for short tasks such as reading the value from a sensor. However, long tasks such as moving to a specific location should be asynchronous. This allows for many operations to be run in parallel and enables a timeout system to be put in place such that commands that do not receive a timely response are sent again.

This communications protocol is very robust. The first safeguard, as previously discussed, is checksums. This ensures that only commands that are received intact are acted upon. A corrupted command could be disastrous. It could instruct an actuator to move into a dangerous position, cause a collision, stall a motor, or tip a piece of lab ware, spilling

chemicals. For example, if a bit were flipped in the command instructing the *Gripper* to return to its zero position, the target could be altered to a much lower position. This could cause the *Gripper* to collide with labware underneath it, or could tip over labware the next time the *OT-2* moves, thinking that the *Gripper* is safely in the home position. This is solved by checksums. In this case, the checksum calculated from the message would not match the checksum included in the message and the command would not be executed.

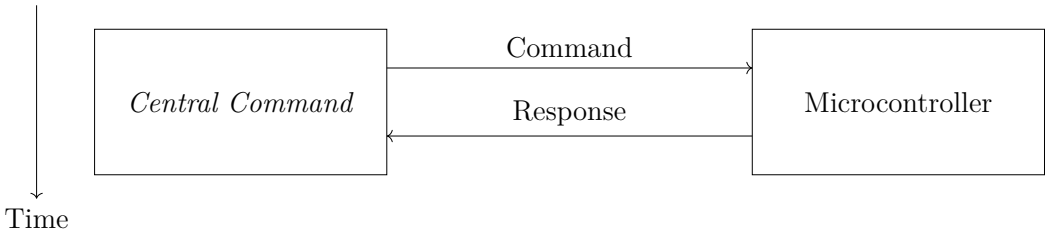
This checksum scheme is an error detecting code, meaning that most errors can be detected, but none can be corrected. It would also be possible to use an error correcting code, such as the Hamming code [149]. Such a scheme can detect errors if many perturbations have occurred, and can even correct errors if a small number of perturbations arise. However, given that the UART connection is fairly reliable, meaning errors are rare, an error detecting code was used for simplicity and efficiency.

Another safeguard that makes the protocol robust is the timeout functionality. If a response has not been received within a specified time period, the command is sent again. In the unlikely event that a packet is lost, which could happen if the microcontroller has just been reset, the *Central Command* module will send the packet again. However, it will not retry indefinitely. After the third attempt, an error will occur. When this happens, there is likely a hardware problem, so the error will alert a human to resolve the error and troubleshoot the connection. Since commands may be sent more than once, it is important that they be idempotent. If the command truly is received multiple times, the second command should have no effect. Commands that toggle a boolean value or move relative to the current position are dangerous. Instead, the command should set the boolean value in the first example and set the absolute position in the second case.

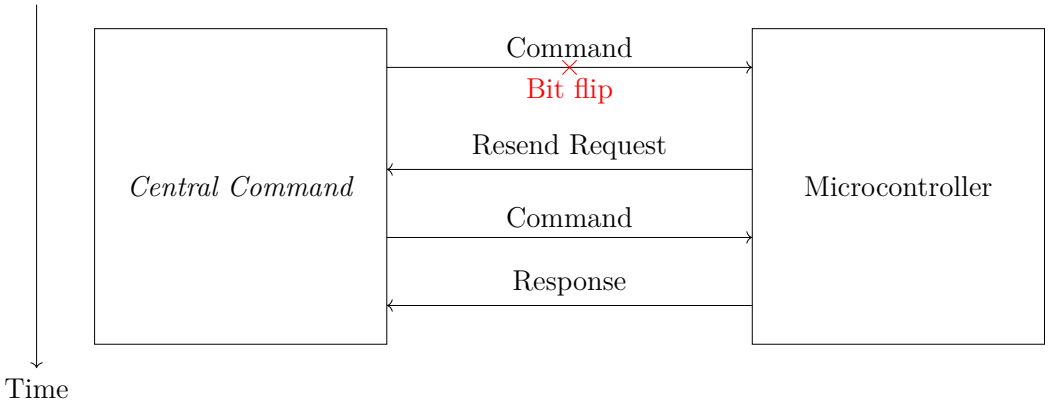
A final safeguard that could be added is to uniquely identify each command and response. In the current implementation, it is possible that a request times out and is sent again. However, the microcontroller would have responded to the initial request given more time. The microcontroller responds to both requests, but the main computer only considers the latest to be valid. From its perspective, the first command timed out and is invalid. The main computer sends a third command before the second response is received and interprets the second response as the response to the third command. The simple addition of message identifier would solve this problem. The main computer would see that the first response received corresponds to the command that timed out. It would discard this response knowing that the response to the second, valid command is forthcoming.

With these safeguards, there are multiple possible outcomes. The most straightforward is that no error occurs. In this case, the command and response are delivered correctly, as can be observed in Figure 4.70a. If a bit flip or some other perturbation occurs during

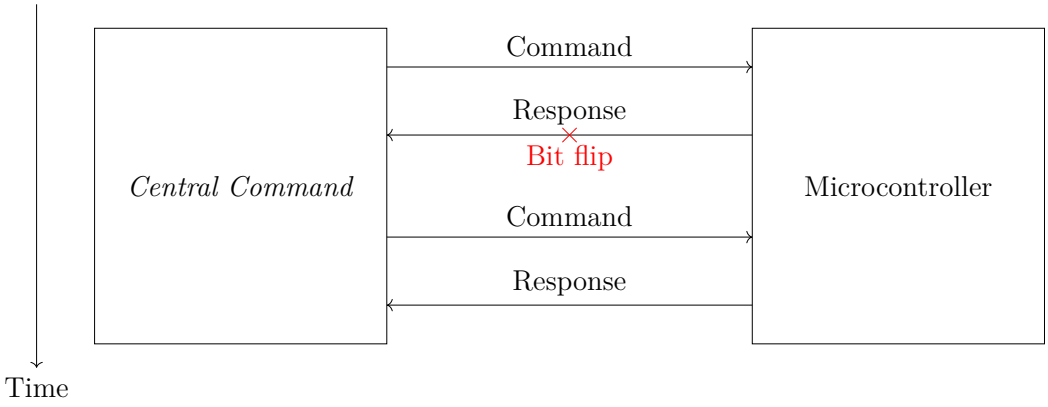
the transmission of the command, the microcontroller will detect this and request that the command be resent, as presented in Figure 4.70b. A similar situation arises if an error occurs during the transmission of the response. This time, it is the responsibility of the *Central Command* software to notice the incorrect checksum and resend the command, as shown in Figure 4.70c. Finally, there are two series of events that can lead to a timeout: the command message can be lost (Figure 4.70d) or the response message can be lost (Figure 4.70e). Each case is identical from the point of view of the *Central Command* software, which will have sent a command without ever receiving a response. In this situation, the command will be sent again.



(a) Best case scenario. There are no errors and a single command and response is sent.

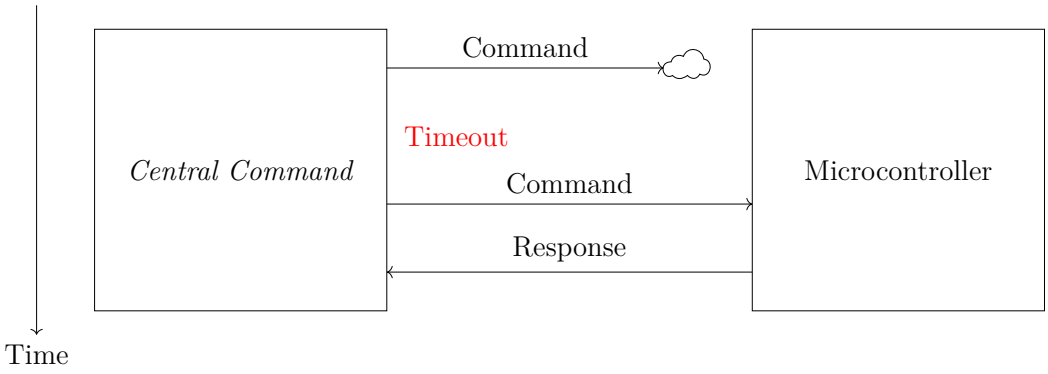


(b) A transmission error occurs during the transmission of the command. The microcontroller will notice a checksum mismatch and send a response requesting that the command be resent.

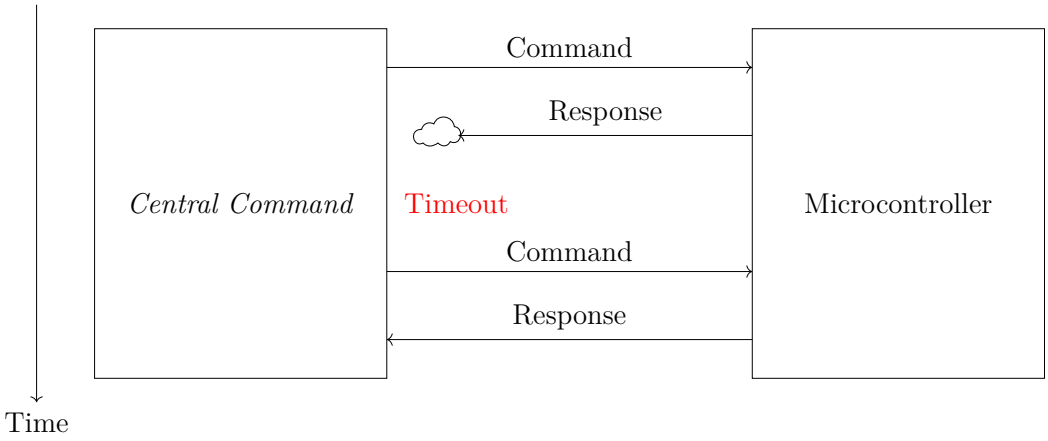


(c) A transmission error occurs during the transmission of the response. The *Central Command* software will notice a checksum mismatch and resend the command. Here, it is very important that the command be idempotent.

Figure 4.70: Diagram of messages between the *Central Command* module and the microcontroller for various scenarios.



(d) The command is never received by the microcontroller. The communication lines could be disconnected or the microcontroller could be powered off. After a certain timeout has elapsed without receiving any response, the *Central Command* software will send the command a second time.



(e) The response is never received by the microcontroller. From the *Central Command* software's point of view, this is the same situation as (d). After a certain timeout has elapsed without receiving any response, the *Central Command* software will send the command a second time. Here, it is very important that the command be idempotent.

[

Diagram of messages between the *Central Command* module and the microcontroller for various scenarios.]Diagram of messages between the *Central Command* module and the microcontroller for various scenarios. (cont.)

4.6.1.2 Asynchronous Acceleration-Capable Stepper Motor Library

Controlling the speed of a stepper motor from a microcontroller is trivial. In a loop, send a pulse to the step pin of the stepper motor driver and add a delay corresponding to the stepping period. However, this naive implementation is insufficient for this application in two major ways. First, the library should be asynchronous. The microcontroller should be responsive to commands while the stepper motor is moving towards its destination. Second, the stepper motor should accelerate slowly rather than starting at full speed. This is especially important if the load on the motor is high, such as the Z-axis of the *Gripper*.

This library addresses these two major problems, as well as including many more features, such as controlling whether the stepper motor driver is enabled (if the stepper has holding torque [150]), controlling the micro stepping, controlling the soft axis limits, translating real-world units such as meters or degrees into the internal units of steps, and reading the TMC2209 [97] diagnostic pin.

§1 Asynchronous operation The asynchronous behaviour of this library is implemented by using an Mbed [102] `Timeout` in order to schedule the next motor step rather than using a blocking delay. This uses the microcontroller's interrupt system in order to call a specific function in an interrupt service routine [151]. Thus, this callback may not use any blocking code, but it is sufficient to pulse the step pin. Every time this function is executed, it must calculate the delay until the next step based on the current desired speed and schedule the next step.

§2 Acceleration The goal of the acceleration feature is to enable movements with constant acceleration. The common trapezoid trajectory planning strategy [152]–[156] is used such that the motor will first be accelerated until the maximum speed is met, then it will be decelerated so that the motor will come to a stop exactly when the target step is reached. The acceleration, characteristic trapezoidal velocity, and displacement traces for a simulated exemplary movement are visible in Figure 4.74.

In order to control the trajectory of a stepper motor with constant acceleration, the period Δt of each step must be calculated. This has units seconds per step and is equivalent to the inverse of the angular velocity ω with units steps per second. It should be noted that all calculations internal to the library use stepper-specific units: steps, steps per second, and steps per second per second. The public interface to the library uses real-world units (§4); however these are immediately translated to internal units. Figure 4.71 shows how the velocity should change from one step to another in order to achieve constant acceleration.

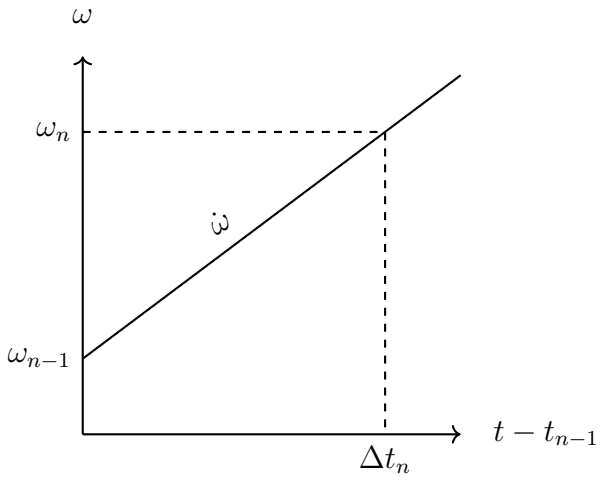


Figure 4.71: Change in angular velocity ω from one step $(n-1)$ to the next (n) with constant acceleration $\dot{\omega}$. t_{n-1} is the time at which step $n-1$ took place and Δt_n is the delay between steps $n-1$ and n, the desired stepping period.

If the angular velocity of the previous step is ω_{n-1} and the angular velocity of the current step is ω_n, then for a constant acceleration $\dot{\omega}$, the delay between step $n-1$ and step n is Δt_n.

A common approach from here is to integrate this slope over Δt_n in order to find the expression $d = \frac{1}{2}\dot{\omega}\Delta t_n^2 + \omega_n \Delta t_n$ where d is the displacement [155], [156] (this may be recognizable as a common equation for constant acceleration). Alternatively, Figure 4.71 leads to Equation 4.1, another common equation for constant acceleration. Rearranging and solving, expression Equation 4.4 is obtained. Both of these approaches lead to solutions for Δt_n involving square roots. This instruction is extremely expensive compared to other arithmetic instructions and will introduce stuttering if computed in real time on modest microcontrollers. It is generally agreed that real-time trajectory planning for stepper motors is not possible on low-end microcontrollers without the use of approximations [155], [156].

$$\omega_n = \omega_{n-1} + \dot{\omega}\Delta t_n \tag{4.1}$$

$$\frac{1}{\Delta t_n} = \omega_{n-1} + \dot{\omega}\Delta t_n \qquad \because \omega = \Delta t^{-1} \tag{4.2}$$

$$1 = \Delta t_n \omega_{n-1} + \dot{\omega}\Delta t_n^2 \tag{4.3}$$

$$\Delta t_n = \frac{-\omega_{n-1} \pm \sqrt{\omega_{n-1}^2 + 4 \cdot \dot{\omega}}}{2 \cdot \dot{\omega}} \tag{4.4}$$

This problem is often overcome by approximating the square root using the Newtonian expansion or by pre-computing and storing a table of stepping periods [155], [156]. However, a different approximation is used for this library. Instead of assuming that the velocity changes from ω_{n-1} to ω_n over Δt_n, it is assumed that this change takes place over Δt_{n-1}. In fact, this was the assumption made originally before it was proven incorrect by simulation. In practise and in simulation (Figure 4.74), this approximation is very good and more than sufficient for this application. Under this assumption, it is possible to find the expression Equation 4.9 for Δt_n as a function of Δt_{n-1} and $\dot{\omega}$ that uses only multiplication, division, and addition (squaring is equivalent to multiplication). This expression was independently derived from Figure 4.71 before it was discovered to be equivalent to Ranade [154] and Xu [157].

$$\omega_n = \omega_{n-1} + \dot{\omega}\Delta t_{n-1} \tag{4.5}$$

$$\frac{1}{\Delta t_n} = \frac{1}{\Delta t_{n-1}} + \dot{\omega} \cdot \Delta t_{n-1} \qquad \because \omega = \Delta t^{-1} \tag{4.6}$$

$$1 = \Delta t_n \cdot \left(\frac{1}{\Delta t_{n-1}} + \dot{\omega} \cdot \Delta t_{n-1} \right) \tag{4.7}$$

$$\Delta t_n = \left(\frac{1}{\Delta t_{n-1}} + \dot{\omega} \cdot \Delta t_{n-1} \right)^{-1} \tag{4.8}$$

$$\boxed{\Delta t_n = \frac{\Delta t_{n-1}}{1 + \dot{\omega} \cdot \Delta t_{n-1}^2}} \tag{4.9}$$

The procedure for the library is thus to iteratively take a step, calculate the period for the next step, and schedule the entire process again until the target step is reached. This scheme is presented in the form of a flowchart in Figure 4.72.

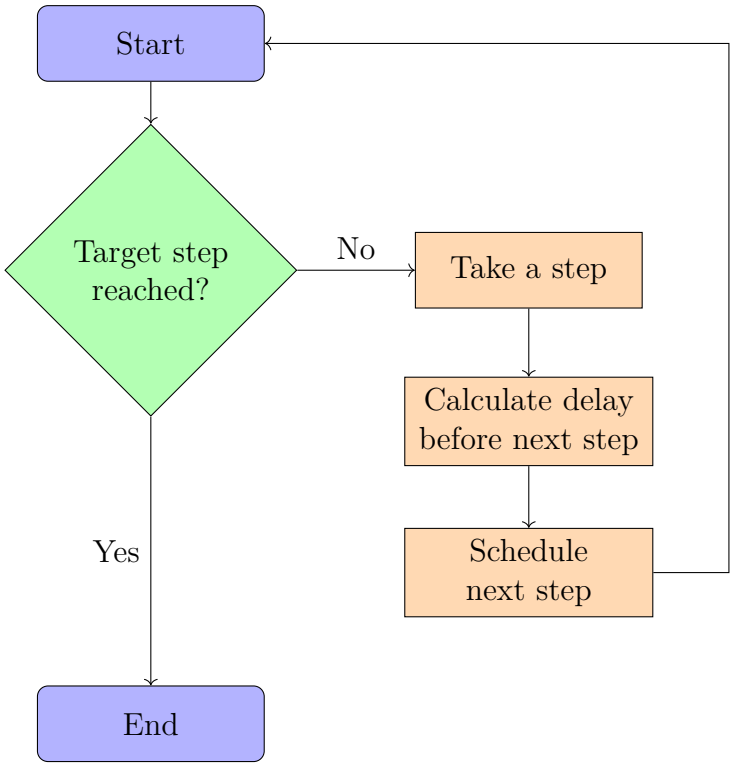


Figure 4.72: Flowchart of the logic to take and schedule steps until the destination is reached.

These equations work well once the motor is already moving, but they break down when scheduling the initial step. In this case, the previous stepping period Δt_{n-1} is not well-defined. To overcome this, a different equation is used to queue the first step. The velocity slope from Figure 4.71 is integrated to find another standard equation for acceleration, this time without the previous step period, which is presented in Equation 4.13. Since the goal is to calculate the delay of the first step to start the motor from a stop, the initial velocity is replaced with zero and the displacement is replaced with 1 step. After rearranging, it is possible to find the solution of Equation 4.15. This equation includes a square root although it was previously mentioned that this operation should not be used. The square root should not be used while the motor is running as this expensive calculation can cause stuttering at high speeds. However, calculating the square root once before the motor has started moving is acceptable.

$$d = \int_0^{\Delta t_n} \omega(\tau) d\tau \qquad \tau = t - t_{n-1} \tag{4.10}$$

$$d = \int_0^{\Delta t_n} \dot{\omega}\tau + \omega_{n-1}\, d\tau \tag{4.11}$$

$$d = \left[\frac{1}{2}\dot{\omega}\tau^2 + \omega_{n-1}\tau\right]_0^{\Delta t_i} \tag{4.12}$$

$$d = \frac{1}{2}\dot{\omega}\Delta t_n^2 + \cancel{\omega_{n-1}}^{0}\Delta t_n \tag{4.13}$$

$$\Delta t_n^2 = \frac{2\cancel{d}^{1}}{\dot{\omega}} \tag{4.14}$$

$$\boxed{\Delta t_n = \sqrt{\frac{2}{\dot{\omega}}}} \tag{4.15}$$

While recalculating the stepping period, two additional steps are required: to constrain the velocity below the configured maximum velocity and to begin deceleration at the appropriate step. Limiting the velocity is easily accomplished by updating the current stepping period if it is less than the minimum period (the inverse of the maximum velocity), as shown in Equation 4.17. Constraining the velocity in this fashion implements the constant speed, zero acceleration region between the acceleration and deceleration of the trapezoid trajectory control strategy [152]–[156]. Here, the left arrow is used as the assignment operator (`=` in most programming languages) so as not to misuse the symbol for mathematical

equality. For example, $x = x + 1$ is a paradox in mathematics, but $x \leftarrow x + 1$ is valid and common in programming.

$$\Delta t_{\min} \leftarrow \frac{1}{\omega_{\max}} \tag{4.16}$$

$$\Delta t_n \leftarrow \max(\Delta t_n,\ \Delta t_{\min}) \tag{4.17}$$

In order to determine if the motor should be decelerated, the number of steps remaining before the target is reached must be compared to the number of steps required to come to a stop. If the desired number of steps is d_{target}, then the number of steps remaining is $d_{\text{remaining}} = d_{\text{target}} - i$. In this case, there are multiple steps. Figure 4.71 holds, but the subscripts n and $n-1$ used for two subsequent steps are replaced with i for the initial step and f for the final step. In this way, it is possible to derive Equation 4.19 and Equation 4.20 in the same fashion that Equation 4.1 and Equation 4.13 were derived. By isolating τ and combining, it is possible to find the standard equation of constant acceleration presented in Equation 4.21. This equation that is independent of time is required because there are now multiple steps; solving for the time required to stop does not help to calculate individual step periods.

$$\tau = t_f - t_i \tag{4.18}$$

$$\omega_n = \omega_{n-1} + \dot{\omega}\tau \tag{4.19}$$

$$d = \frac{1}{2}\dot{\omega}\tau^2 + \omega_i \cdot \tau \tag{4.20}$$

$$2d\dot{\omega} = \omega_f^2 - \omega_i^2 \tag{4.21}$$

Since it is a stop, the final velocity $\omega_f = 0$, and since the deceleration starts at the current step, the initial velocity ω_i is the current velocity ω_n. Rearranging, the result of Equation 4.24 is found. Here, the negative sign indicates that the acceleration should be negative since this is a deceleration. In this library, the same value is used for acceleration and deceleration. Only the sign is flipped. A deceleration-aware acceleration value a' is created based on the stage of the movement (Equation 4.25), and this value is used when calculating the stepping period rather than the desired, strictly positive value a. Thus, the "Calculate delay before next step" step from Figure 4.72 is performing a sequence of multiple tasks enumerated in Figure 4.73.

$$2\dot{\omega} d_{\text{stop}} = \cancelto{0}{\omega_f^2} - \omega_n^2 \tag{4.22}$$

$$2\dot{\omega} d_{\text{stop}} = \frac{-1}{\Delta t_n^2} \tag{4.23}$$

$$\boxed{d_{\text{stop}} = \frac{-1}{2\dot{\omega}\Delta t_n^2}} \tag{4.24}$$

$$\dot{\omega}' = \begin{cases} \dot{\omega} & d_{\text{stop}} < d_{\text{remaining}} \\ -\dot{\omega} & \text{otherwise} \end{cases} \tag{4.25}$$

This finalises the equations of the stepper motor library. The classic curves of the trapezoid trajectory planning strategy [152]–[156] for a simulation of this library are presented in Figure 4.74. The staircase-like velocity of Figure 4.74b is achieved by calculating the slope of the discrete stepper positions of Figure 4.74c using `numpy.gradient` [158]. A curve similar to Figure 4.74c is obtained when inspecting the step pin going to the stepper motor driver and recording the times of the pulses, validating that this method works in practise as well.

This library calculates the stepping period on-the-fly. This has the advantage of being very dynamic; any configuration setting can be updated while the motor is running and the library will adjust the trajectory accordingly. It is possible to update the acceleration value, the maximum speed, or even the target position while the motor is turning. Every segment of the trapezoidal trajectory is handled dynamically in a way that is easy to reason about (Figure 4.73). For example, if there is insufficient distance to reach the maximum velocity before the destination is reached, then this portion is skipped. Furthermore, this method eliminates the complexity and space required to calculate and store pre-computed acceleration ramps.

§3 Soft axis limits The stepper motor library implements soft axis limits that can be used to ensures that the stepper motor is not moved outside of its designated range. For example, the *Gripper* module (Section 4.1) sets the axis limits such that the head cannot be moved above the Z-axis limit switch or below the end of the linear rail. If the library receives an instruction to move outside of these limits, the error `OUT_OF_BOUNDS` is returned (§6). The limits are initially set to ∞ and $-\infty$ so that they have no effect unless explicitly updated.

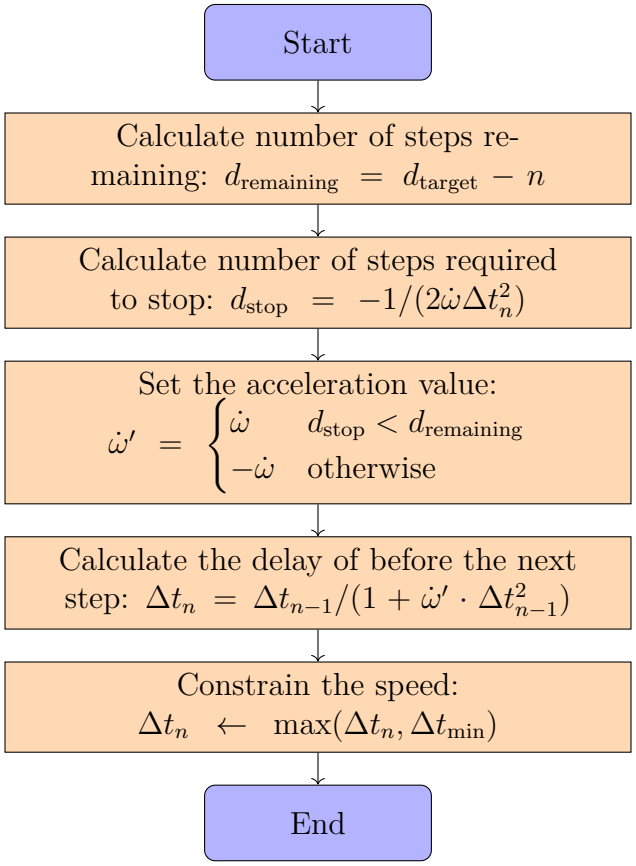


Figure 4.73: Sequences of steps to update the stepping period for the next step, taking the maximum velocity and deceleration into account.

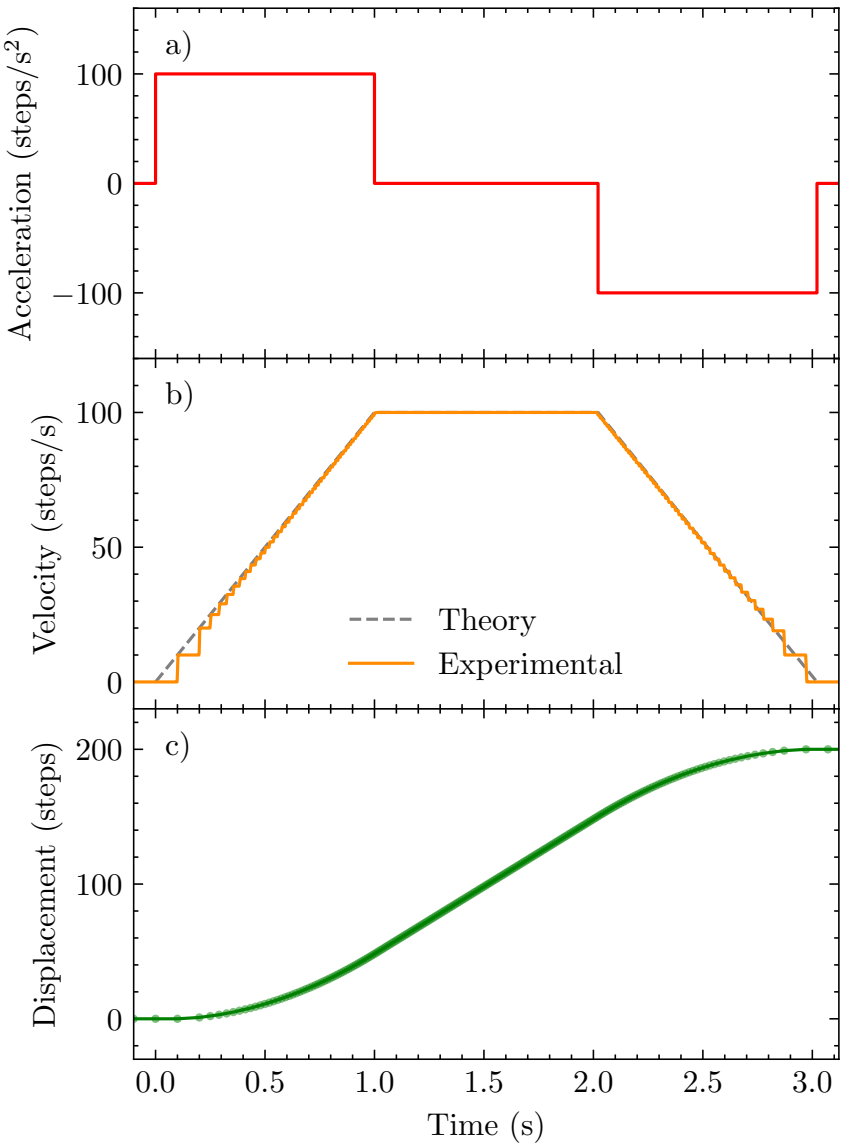


Figure 4.74: A simulation of the constant acceleration equations for a displacement of 200 steps with an acceleration of 100 steps/s^2 and a maximum velocity of 100 steps/s showing a) the acceleration, b) the velocity, and c) the displacement as a function of time. The theoretical gray dashed velocity is expected for this acceleration. However, the discrete nature of stepper motors results in the orange experimental signal. The individual steps are shown by the markers of c) and the staircase pattern of b), where it is possible to see that steps are more frequent for higher velocities.

§4 Real-world units The library has a notion of real-world units that are different from the internal, stepper-specific units based around an integer number of steps. For example, the real-world units are millimeters for the Z-axis and revolutions (after the worm gear) for the A-axis of the *Gripper* module (Section 4.1).

The public interface to the library entirely uses these real-world units, while the internal calculations in the library use steps. Therefore, every interaction with the library must translate between the two. This is accomplished by specifying a conversion ratio when instantiating the library. This often includes the steps per revolution of the stepper motor and any gears, pulleys, or lead screws. For example, the calculation of steps per millimeter of the *Gripper* Z-axis involves the motor's steps per revolution, the timing belt between two pulleys, and the lead screw, as seen in Equation 4.26. The calculation of steps per revolution of the *Gripper* A-axis involves only the motor steps per revolution and the worm gear ratio as seen in Equation 4.27.

$$Q_{\text{steps/mm}} = \frac{200\,\text{steps}}{\text{revolution}} \cdot \frac{40\,\text{teeth}}{20\,\text{teeth}} \cdot \frac{\text{revolution}}{8\,\text{mm}} \tag{4.26}$$

$$Q_{\text{steps/revolution}} = \frac{200\,\text{steps}}{\text{revolution}} \cdot 60 \tag{4.27}$$

§5 Public Interface The public methods of the stepper motor library are enumerated in Table 4.14.

Table 4.14: Public methods of the stepper motor library class.

Method Name	Description
`set_target`	Set the destination in real-world units.
`set_target_relative`	Set the destination relative to the current position in real-world units. Not to be used directly from a UART command; not idempotent.
`get_accel`	Read the current acceleration.
`set_accel`	Update the acceleration.
`get_max_speed`	Read the current maximum speed.
`set_max_speed`	Update the current maximum speed.
`start_infinite_rotation`	Start an infinite rotation without a destination. Useful for homing sequences.
`stop`	Immediately stop all movement.
`enable`	Enable the stepper motor driver, applying holding torque and enabling the motor to be moved.
`disable`	Disable the stepper motor driver, disengaging holding torque and disabling the motor from being moved.
`is_done`	Check if the motor has reached its destination.
`block_until_done`	Wait for the movement to complete, effectively turning the asynchronous library into a synchronous library.
`current_position`	Read the current position in real-world units.
`max_pos`	Read the current axis upper limit.
`min_pos`	Read the current axis lower limit.
`set_min_pos`	Update the current axis upper limit.
`set_max_pos`	Update the current axis lower limit.
`diag`	Read the diagnostic pin of the stepper motor driver [97].
`index`	Read the index pin of the stepper motor driver [97].

§6 Error Conditions Currently, the only error condition implemented in the stepper library's status enumeration type [101] is `OUT_OF_BOUNDS`, as can be seen in Table 4.15. This enumeration is returned from the `set_target` and `set_target_relative` methods to indicate whether the movement has started successfully. It is possible that the stepper motor driver will enter an error state and activate the diagnostic pin during the movement. However, due to the asynchronous nature of this library, this cannot be known when `set_target` returns and is thus not implemented in this enumeration. It is the

responsibility of the library's caller to check the diagnostic pin of the stepper motor driver in the loop used to check if the movement has finished.

Table 4.15: Variants of the stepper library status enumeration type.

Variant	Value	Description
`OK`	0	There is no error and the movement has started.
`OUT_OF_BOUNDS`	1	The instruction would cause the motor to move outside its soft limits (§3).

4.6.1.3 AS5600 Magnetic Induction Rotary Encoder Library

The magnetic induction rotary encoder library is responsible for reading the value of an AS5600 sensor over the I^2C bus (Section 4.1.2.3). The library includes a zero offset and direction to improve the ease of use. It is possible to specify some partial rotation to be the zero angle, which is useful to compare two different sensors where the magnets may have been installed slightly differently. The direction is also useful if the value of the sensor increases contrary to intuition. For example, if the clockwise direction is intuitively "positive" but the sensor was installed in such a way that a clockwise rotation results in a smaller reading, it is possible to correct this by setting `reverse=true`. These two options allow such settings to be applied globally in a single location rather than including `- zero` and `* -1` throughout the code.

The library also tracks the number of full rotations so that absolute positioning above 360° is possible. Once the angle passes half of a full rotation, the number of rotations is incremented.

The methods of the `AS5600` class are given in Table 4.16.

Table 4.16: Public interface of the magnetic induction rotary encoder library.

Method	Description
`read()`	Read the value from the sensor, applying the full revolutions, zero offset, and direction.
`read_raw()`	Read the raw value from the sensor.
`is_magnet_present()`	Check if a magnet is present by verifying the magnetic field strength.

There is also a multiplexer-aware subclass of this library. As specified in Section 4.1.2.1, the AS5600 has a fixed I²C address. Thus, to address multiple sensors on a single bus, an I²C multiplexer is required. The multiplexer connects to the main I²C bus of the microcontroller and connects this to one of its multiplexed downstream buses. Sending a byte with the index of a downstream bus to the address of the multiplexer will switch connect the main I²C bus with that downstream bus. Any targets[1] on this downstream bus will then be addressable as if they were on the main bus.

The multiplexer-aware variant of the rotary encoder library has the same public interface as the original, except that the multiplexer index is an additional argument to the constructor. This variant works by sending its own index to the multiplexer when any method is called before calling the original implementation.

4.6.2 Libraries for use in Opentrons protocols

This section details the libraries developed to facilitate the creation of *Opentrons* protocols. Section 4.6.2.1 describes the library for working with stocks, while Section 4.6.2.2 prevents collisions with the *Gripper*.

4.6.2.1 Liquid Library

The purpose of the liquid library is fourfold. First, the library stores all the settings required in order to manipulate all chemicals used in the ATPE process. Different chemicals with different viscosities will need varying aspiration and dispensing rates, blow out, and air gaps. For example, dextran is very viscous and needs to be dispensed very slowly, needs an air gap, and needs multiple blow outs. Water, on the other hand, can be aspirated at full speed with no air gap. The second purpose of the library is to track the remaining liquid of each stock in use. The stocks are tubes of unmixed chemicals that the *OT-2* will combine during the ATPE process. Tracking the volume remaining in these tubes is essential to ensure that there are sufficient quantities of all stocks to complete a program before it is started and to insert the tip of the pipette an appropriate distance into the tube. The tubes are quite tall, so if the liquid is always aspirated from the bottom, the entire tip and part of the pipette will be submerged if the tube is full, which is not desirable. Third, the library is able to transfer quantities above the capacity of the pipette by making multiple

[1] This text uses the terms “controller” and “target” rather than the politically problematic “master” and “slave” used in the original I²C specifications.

smaller transfers. Fourth, as the library is aware of the liquid being transferred, the tip of the pipette is only changed when the liquid is changed to reduce the wasted tips.

Each stock should have a well-defined position in the *OT-2*. Usually, two tube racks are used for stocks, each having 10 slots. The tip rack position and slot number must be specified for each stock when instantiating the liquid library. An example of this is shown in Listing 1, where a tube rack is defined in deck position 11 and the location of the tube containing dextran (with codename `DX`) is said to be the top left slot of that tube rack when setting up the liquid library class.

```
stocks_tuberack_1 = protocol.load_labware(
  'custom_50ml_tube_rack_v2',  # Identifier of this kind of tube rack
  location=11,  # The location in the \ot{} (1 of 11 deck positions)
)
LiquidLibrary(protocol=protocol, locations={
  'DX': stocks_tuberack_1['A1'],  # A is the first row, 1 is the first column
})
```

Listing 1: Instantiating and providing the stock locations to the liquid library.

After the positions are defined, the liquid library can be used to transfer stocks by their name, rather than by their location. The line `pipette.transfer(source=stocks_tuberack_1['A1'], ...)` is replaced with `liquid_library.transfer(liquid='DX', ...)`. This has numerous advantages. The locations of stocks are stored in a central location rather than duplicated throughout the program. The tracked volume of the stock is updated to reflect that some liquid has been taken. Finally, this enables the other advantages of the liquid library, such as volumes above the pipette capacity or aspirating in the correct position based on the liquid level.

§1 Encoding the liquids and properties The available liquids and their corresponding properties are stored in a `.csv` (comma-seperated values) file inside the *OT-2*. This is a simple file format for encoding tabular data in plain text. The advantage of this format is that it can easily be read and written from code and from any spreadsheet software. This is a compromise between an intuitive user experience for non-developers and simplicity and ease of implementation. The user can change the properties of a liquid using a familiar spreadsheet interface. However, it may not be obvious how to transfer this file to the *OT-2* after having edited it in a spreadsheet software. Nevertheless, this is a reasonable compromise as creating a GUI for this small task would not be worthwhile.

The properties that should be encoded for each liquid are its name, the aspiration and dispense speeds ($\mu L\,s^{-1}$), the air gap volume (µL), the blow out volume (µL), the blow

out count, and whether to touch the tip to the side of the container to remove any drops. When using the `transfer` method, all of these parameters are automatically used.

§2 Volume tracking The volume of each liquid is stored in a `.json` (JavaScript object notation) file inside the *OT-2*. Each time a stock is transferred, the amount aspirated is subtracted from the value in this file.

The *OT-2* simulates every protocol before it is run. This means that if there is insufficient liquid to fulfill a given protocol, this can be detected during the simulation phase even before the protocol is run. However, it is important that this simulation mode be detected and that the updated volumes are not saved to disk, as these are not truly aspirated during the simulation.

When the stocks are manually replenished, the volumes in this file must be updated. In order to facilitate this task, the Opentrons app [92] has been updated such that these volumes can be viewed and updated from the GUI. This app is open source, so simple modifications such as this are possible.

Before the protocol is run, the volumes can be updated from the form that was added to the "Protocol" tab. This form is presented in Figure 4.75. While the protocol is running, the volumes are displayed and updated in real-time in the "Run" tab. This is accomplished by running a WebSocket [159] (a two-way real-time communications protocol built on TCP) server from the liquid library. When a client connects to the server or the volumes file is saved to disk, the volumes are published to the new client or to all clients. This enables real-time updates of the volumes of the stocks, which can be seen decreasing as the *OT-2* aspirates liquids. When a message is received, it is parsed as a JSON-encoded dictionary of volumes and saved to disk. A locking mechanism is used to ensure that the file cannot be updated by other processes while a protocol is running.

4.6.2.2 *Gripper*-Aware Pipette

The `SafePipette` library is a wrapper around the default Opentrons `InstrumentContext` class [160] that prevents the *OT-2* from moving if the *Gripper* is not in the home position. This prevents collisions that could occur when the head of the *OT-2* is moved when the *Gripper* is in the lowered position. Thus, the `SafePipette` wrapper should always be used when the *Gripper* is installed. This was developed with the guidance of Cousins who contributed to a GitHub issue on this subject [161].

All instructions that move the head of the *OT-2* originate from a method on an `InstrumentContext` instance, since all movements are relative to the tip of the pipette [161].

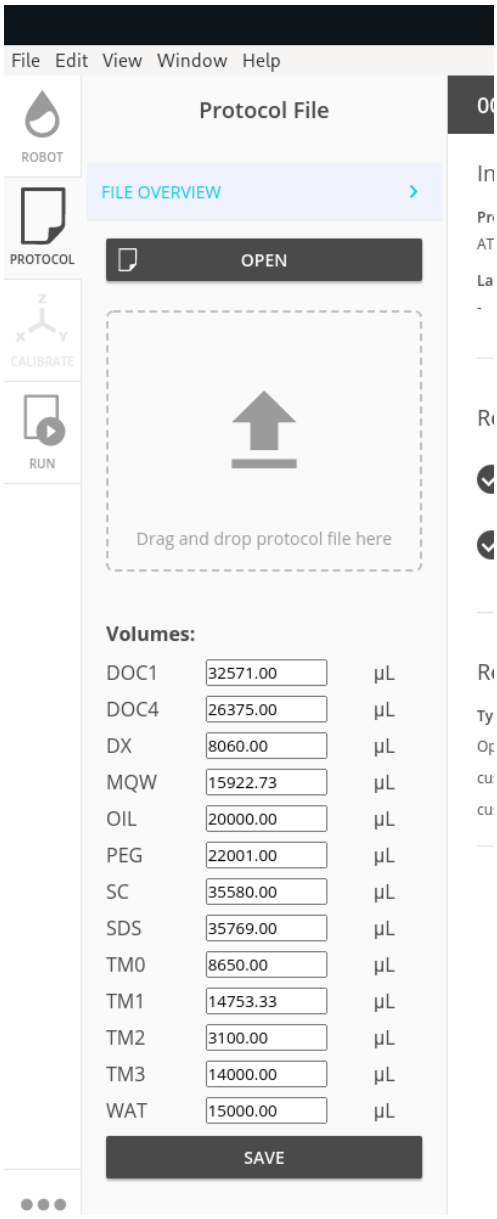


Figure 4.75: Screenshot of a modified version of the Opentrons app to support updating the volumes from the GUI.

This integrates with the *Gripper* library to send check if the limit switch is active before every movement. The goal is to seamlessly replace the `InstrumentContext`. Thus, the public interface is not affected. The `__getattr__` and `__setattr__` special methods [162] are used to intercept any methods on the main class that would cause the *OT-2* to move.

4.6.3 Manual control interface

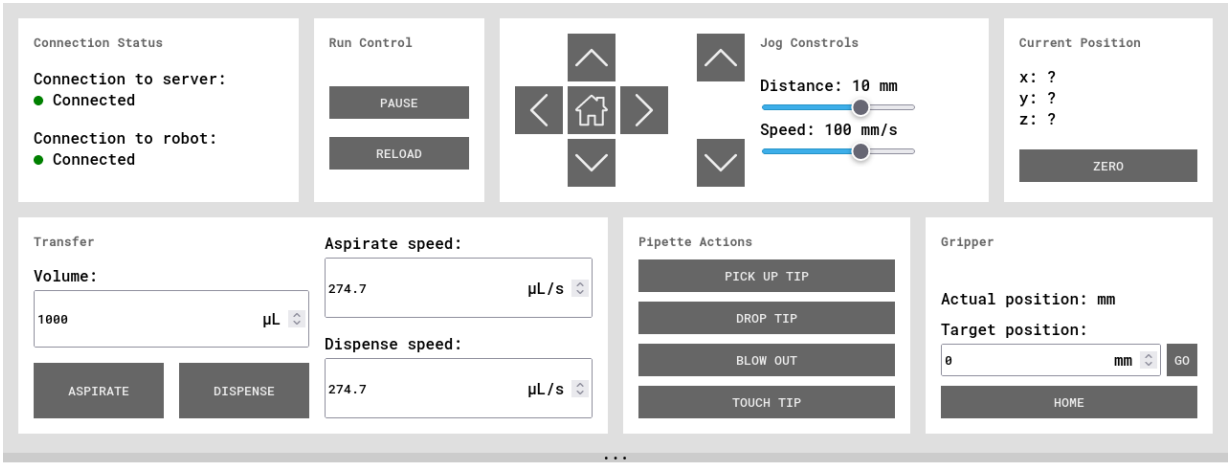
The *Manual Control Interface* is a user interface designed to interact with the *OT-2* and custom modules for the purposes of debugging, experimentation, and module development. This interface was born out of the desire to manually control the position of the robot, a notable omission from the official *Opentrons* app, but has since grown to include many more features. This app has proven invaluable during the development of the custom modules.

A screenshot of the *Manual Control Interface* is presented in Figure 4.76, which shows the many features included designed to efficiently debug and develop the automated ATPE system. On the top, there is a grid of tiles, each providing functionality with a specific theme. The purpose of each tile is given in Table 4.17. On the bottom, there is an interactive command-line interface, which is described in Section 4.6.3.1.

4.6.3.1 Interactive Command-Line Interface

The interactive command-line interface of the *Manual Control Interface* is a Python REPL where commands are run directly on the *OT-2*. Here, it is possible to run arbitrary Python commands and to view the output and return value. For example, it is possible to perform arithmetic or to print the current time. This interface additionally has access to certain protocol variables, such as the `InstrumentContext` of each pipette [160], the `ProtocolContext` object [160], the liquid library (Section 4.6.2.1), and all custom modules. Thus, virtually all functionality accessible through a protocol is possible through this interface. This is invaluable when developing the Python library for a custom module as it is possible to quickly and interactively test a new method in the *OT-2* while having access to all protocol functions.

The REPL has some additional features compared to certain simple command-line interfaces. First, commands are coloured using Python syntax highlighting, which is useful to quickly identify mistakes. It is possible to see that the `print` function is coloured blue and strings are coloured green in Figure 4.76. Furthermore, commands, return values, messages, and errors are distinguished. A right-facing arrow is shown before commands,




```
❯ locals()                                                                                    02:59:47
❮ {'protocol': <opentrons.protocol_api.protocol_context.ProtocolContext object at 0x7fcf22004e80>, 'ws':   02:59:47
  <websockets.legacy.client.WebSocketClientProtocol object at 0x7fcf217628b0>, 'queue': <Queue at 0x7fcf2177c520
  maxsize=0 tasks=2>, 'vial_opener': <MagicMock id='140527596594800'>, 'tiprack': Opentrons 96 Tip Rack 1000 µL on 11,
  'rack': Custom 15mL vial rack on 1, 'source': Custom 50mL Tube Rack V2 on 4, 'ultimate_answer': 42, 'executor':
  <function handle_long_commands.<locals>.executor at 0x7fcf21780d30>, 'gripper': <MagicMock id='140527596796080'>,
  'loop': <_UnixSelectorEventLoop running=True closed=False debug=False>, 'pipette':
  <central_command.lib.safepipette.SafePipette object at 0x7fcf217cf3d0>}
❯ 2+2                                                                                         02:59:51
❮ 4                                                                                           02:59:51
❯ ultimate_answer                                                                             02:59:54
❮ 42                                                                                          02:59:54
❯ print(sys.version)                                                                          03:00:06
❮ None                                                                                        03:00:06
  3.9.12 (main, Apr 29 2022, 14:45:39)                                                        03:00:06
  [GCC 11.2.0]
❯ Deliberately causing a syntax error                                                         03:00:22
❮ Traceback (most recent call last):                                                          03:00:22
    File "/home/marcel/code/quin/CNT_Automation/machine_interface/real_protocol.py", line 259, in run_repl_command
      'value': str(exec_with_return(command, _locals)),
    File "/home/marcel/code/quin/CNT_Automation/machine_interface/real_protocol.py", line 36, in exec_with_return
      code_ast = ast.parse(code)
    File "/home/marcel/.pyenv/versions/3.9.12/lib/python3.9/ast.py", line 50, in parse
      return compile(source, filename, mode, flags,
    File "<unknown>", line 1
      Deliberately causing a syntax error
                 ^
  SyntaxError: invalid syntax
❯ print('This will be red', file=sys.stderr)                                                 03:00:52
❮ None                                                                                        03:00:52
  This will be red                                                                            03:00:52
❯
```

Figure 4.76: Screenshot of the *Manual Control Interface*.

Table 4.17: The functionality provided by each tile of the *Manual Control Interface* from left to right, top to bottom.

Tile Title	**Function**
Connection Status	Displays the status of the connection to the server and the *OT-2*.
Run Control	Pause the currently running action or reload the *OT-2* protocol.
Jog Controls	Move the head of the *OT-2* by small increments or home the *OT-2*. The distance of these increments and the movements speed are adjustable. When this tile is selected, the arrow keys also move the *OT-2*.
Current Position	Displays the current position of the head of the *OT-2*, or `?` if the *OT-2* has not been homed. The "ZERO" function saves the current position and subtracts it from the displayed position similarly to the zero function of a kitchen scale. This is useful to measure relative distances, like the height of a liquid in a vial after zeroing at the brim.
Transfer	Aspirate or dispense a given volume with a given speed in the current position.
Pipette Actions	Common pipette actions: attach the next tip in the configured tiprack to the pipette, drop the currently attached tip in the trash, blow out the tip to remove residue, touch tip to the edges of the current well to remove drips.
Gripper	Control the height and home the *Gripper* custom module (Section 4.1).

while a left-facing arrow is shown before return values. Messages output by functions like `print` are displayed without an arrow. Errors are presented like return values except that the text and arrow are red. Similarly, messages printed to `STDERR` are also displayed in red text. Each of these cases is visible in Figure 4.76. Every command and message has a corresponding time stamp to keep an execution history. Finally, the list of previously executed commands is saved. Pressing the up arrow will return to the previous command, where it can be modified and rerun. Typing the start of a command and pressing the up arrow will return to the previous command that starts in the same way. This is useful to find a long command where only the start can be remembered. This history is persistent and is not lost by refreshing the browser or restarting the server.

Running arbitrary Python code submitted to a server is usually a big security concern. However, in this instance, the risk is acceptable. First of all, the server is behind a firewall; everyone granted access to the network is trusted. Furthermore, the special *OT-2* protocol is typically only run for short periods of time while the *Manual Control Interface* is actively being used. This is because only one protocol can be run at a time, so this is shut down when not in use. Finally, the interface already provides buttons that move a large liquid handling robot, which can do physical damage when used maliciously. In this case, executing Python code is of little concern. All persons accessing the network must be trusted.

4.6.3.2 Implementation and architecture

The *Manual Control Interface* has three major components: the server, the client, and the *OT-2* protocol. The server is a web server tasked with managing communication between the client and the *OT-2*. The client is a webpage that can be accessed by visiting the address of the server in a web browser. Finally, the *OT-2* protocol is a special *Opentrons* protocol that, instead of executing a sequence of liquid transfers and exiting like a normal protocol, waits for commands to be received from the server.

The reason to implement this interface as a website instead of a native application is that website development is very accessible and so that anyone can access the interface by visiting the server without the need to install software on their computer.

The protocol is written in Python as required by *Opentrons*. The client is written in JavaScript using the *React* framework [163] for easy user interface development. The server is also written in JavaScript using the *Node.js* [164] runtime to be consistent with the client and because WebSockets [159] are easier in JavaScript than in Python.

All messages are sent using WebSockets [159], a two-way real-time communications protocol built on TCP, via the server. Therefore, the latter creates two WebSocket servers: one for the clients and one for the *OT-2*.

Chapter 5

Integration of External Instruments

The *Opentrons OT-2* liquid handling robot and the custom modules developed for it work well for hybrid ATPE (Section 6.1). In this mode, the automated system is able to mix solutions and perform separation. However, a human operator is required to place vials in the centrifuge, to place stocks in the sonicator, and to prepare samples for the spectrophotometer. Thus, an external robotic arm is added so that containers can be transported into and out of the *OT-2*. This arm will interact with these aforementioned external instruments and will play a crucial role in efficient separation by mimicking human motion to aspirate the meniscus (Section 4.3.2.7).

The most important external instrument to integrate into the automated ATPE process is the centrifuge since this is crucial to the separation process. It is only via the centrifuge that the two phases are formed. The sonicator is only required for stock-making and the spectrophotometer is only required for characterisation.

5.1 Robot Selection

Multiple options for a robotic arm were considered. To avoid the safety requirements of industrial robotic arms and to allow humans to work alongside the robot, the search was limited to collaborative robotic arms. These have reduced momentum and additional internal sensors that permit them to be used in the presence of humans, whereas a typical industrial robotic arm moves at high speeds to optimize cycle time and is installed in a cage with extensive access control to protect nearby humans [165].

Initially, the *Universal Robots UR5e* [166], *Fanuc CR-4iA* [167], and *Kuka LBR iiwa 7 R800* [168] collaborative robotic arms were considered. These companies were contacted for quotes. However, while vising the *Waterloo RoboHub* [169] to view a *Universal Robots* arm in person, a fourth option was recommended: the *Franka Emika Panda* [170]–[173]. Eventually, this option was chosen for its lower cost, seventh axis, and recommendation from *RoboHub*. An image of the selected *Franka Emika Panda* robot is presented in Figure 5.1.



Figure 5.1: Image of the *Franka Emika Panda*. Reproduced from [174].

The robot was purchased from the local reseller *Clearpath Robotics* [175]. A special table was fabricated for the arm and associated equipment with the assistance of Roberto Romero. This table securely holds the robotic arm, but also has adapters to hold the *OT-2*, centrifuge, sonicator, and spectrophotometer in place. It is crucial that these elements do not move on the table to enable repeatable positioning for the automated robotic system. A photo of the plan for this table is presented in Figure 5.2.

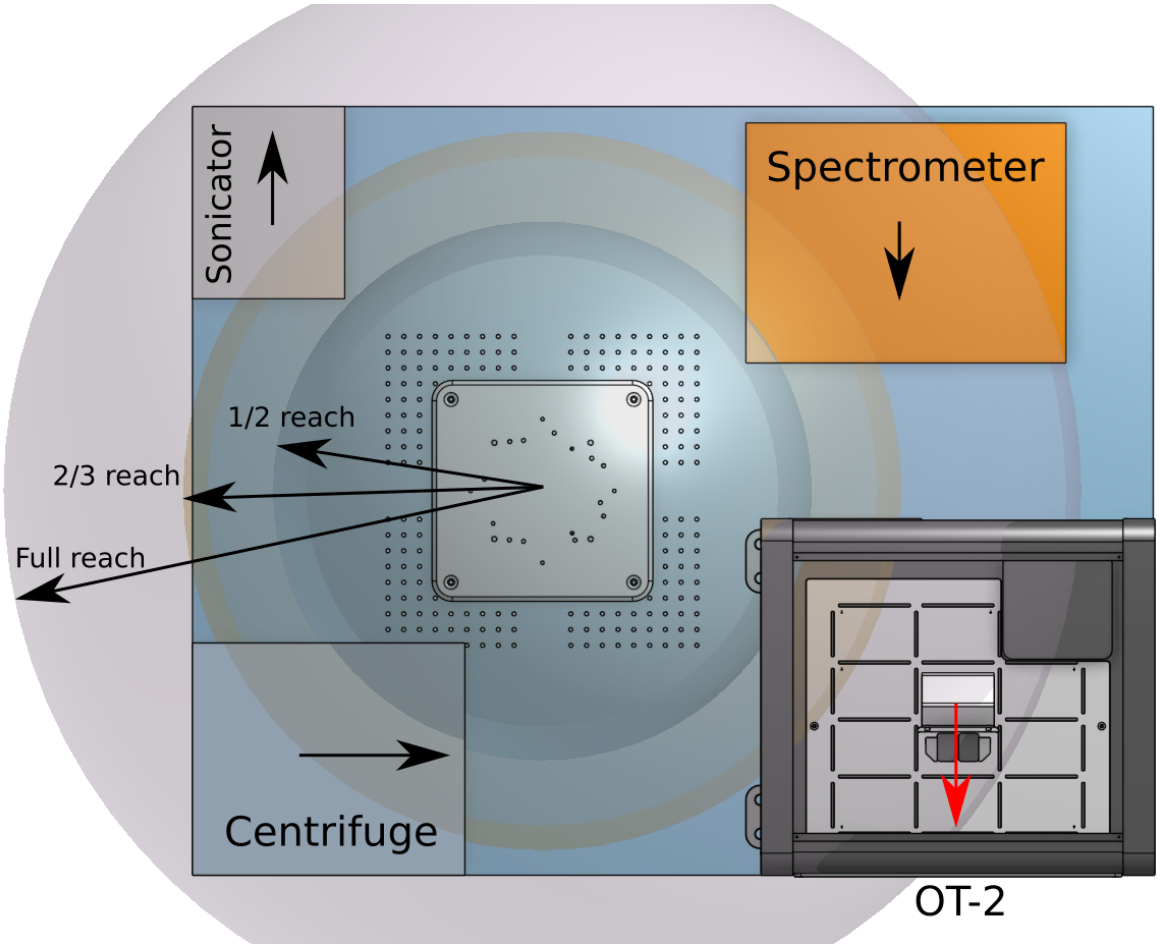


Figure 5.2: Sketch of the planned placement of instruments on the robotics table. Arrows point towards the front of each instrument. The robotic arm is to be installed on the aluminum plate near the centre of the image. Transparent spheres show 1/2, 2/3, and 1/1 of the robot's reach. These guides allow the instruments to be positioned so that the robot performs the majority of its tasks between 1/2 and 2/3 of its full reach, as recommended by the *Waterloo RoboHub*.

5.2 *OT-2* Integration

In order to integrate the *OT-2* into the external robotic arm system, two steps are required. First, small brackets have been 3D-printed that, when bolted to the table, fully enclose the

feet of the *OT-2*, preventing it from sliding on the table. This way, reliable positioning is achieved for the robotic arm. Second, the side panel must be removed to allow the arm to enter and manipulate vials inside the *OT-2*. It was decided to reserve the front of the liquid handling robot for its initial purpose of human access via a hinged door. The robotic arm will enter from the side of the *OT-2* to pick up or deposit vials or to perform separation (Section 4.3).

The coordination of the *OT-2* and robotic arm will be done over TCP. The robotic arm does not have a Python interface. There are robot operating system (ROS), Matlab, and C integrations for an additional hefty fee [176][1]. However, none of these is perfectly suited for this use case. While ROS is written in Python, this framework would not integrate well with the standard Python of the *Opentrons OT-2*.

The plan to enable communication between the *OT-2* and the robotic arm is to develop custom "Apps" for the DESK GUI. This is the interface used to program the robotic arm, and it is possible to develop additional modules. The plan is to use the *RIDE* programming ecosystem [177] to enable TCP communications with the *OT-2*. Protocols for the latter are written in standard Python, so it is straightforward to add networking to this. In fact, it has already been done for certain other modules and for the *Manual Control Interface* (Section 4.6.3).

However, the robotic arm is part of the second phase of this project, which is not yet complete. Thus, no further work has been achieved on these custom *RIDE* apps or on this communication between the *OT-2* and the robotic arm.

5.3 Centrifuge Integration

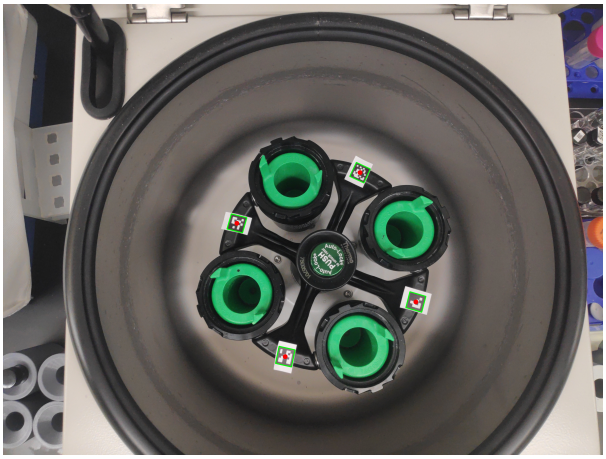
Like the *OT-2*, the centrifuge requires 3D-printed brackets to hold its feet in repeatable locations. Furthermore, the position of the vials in the cups of the centrifuge must be determined. After the centrifuge is run, the angle of the rotor cannot be guaranteed. Such a detection system is required so that the robotic arm can know the positions of the cups of the centrifuge and which vial is in which cup. Thus, a computer vision system is implemented to detect the rotor position after the centrifuge is run. This was deemed the most reliable and unobtrusive (relative to installing an encoder) option to find the rotor angle. A small marker must be placed on each of the arms of the rotor that are recognized by a camera. The plan is to mount a camera above the centrifuge facing downwards. When the lid of the centrifuge is opened after being run, a photo of the markers on the rotor will

[1]The source for the pricing is a confidential quote that cannot be referenced

be taken and computer vision code will detect the markers in the image. The positions of these can be used to determine the angle of the rotor relative to the camera. Since the camera is rigidly mounted, this allows the angle of the rotor to be repeatably estimated.

The camera has not yet been mounted above the centrifuge. However, multiple experiments have been performed holding a cell phone camera above the centrifuge in order to test the viability of recognizing markers in this fashion. The first attempt used small QR codes on each arm. The labels were printed using a label maker so that they could easily be adhered to the centrifuge. Unfortunately, at this small size, the markers were not reliably detected in the image. Therefore, a different type of markers known as ArUco [107], [108] were installed. These have the advantage of a lower resolution. Thus, they can store less data, but are easier to recognize in an image. Whereas QR codes excel at storing data, ArUco codes are superior to uniquely identify a small number of known items, like the arms of a centrifuge.

This new kind of marker is much more reliable. As can be seen in Figure 5.3, all four codes are correctly identified even in the case of an extreme angle in the camera (sub-figure a). This angle is used to test the limits of the ArUco detection. The ArUco detection support in *OpenCV* [132] is used to recognize code in the image. The code is inspired from [107].



(a) Mostly top-down image



(b) Image with exaggerated angle to test the limits of the marker-finding

Figure 5.3: Two images of 4 correctly detected ArUco codes in a cell-phone image of the centrifuge rotor. The red circles and green borders are added by the computer vision software and indicate the centre and outline of the code, respectively. The presence of these annotations proves the successful detection of the codes.

This concludes the steps needed for the centrifuge integration. All other requirements are met by the robotic arm, such as opening and closing the lid and pressing the buttons to start the run.

5.4 Sonicator Integration

The sonicator requires two additions in order to work well in the robotic system. First, like the other instruments, 3D-printed brackets that constrain the feet from sliding on the table should be designed and implemented. Second, a repeatable positioning system for the tubes must be implemented. The solution for manual use was to use pieces of foam with holes large enough for tubes to fit inside to allow the tubes to float above the bottom of the sonicator. This is not conducive to automation since the floating foam pieces can stray from their original position in the water. The only constraint is that the tubes do not touch the bottom of the basin, they do not need to be floating. Therefore, a 3D-printed plate was designed that tightly fits into the top of the basin. This plate has a grid of nine holes that hold the tubes in repeatable positions. An image of this 3D model is presented in Figure 5.4.

These are the only physical augmentations required to integrate the sonicator into the robotic system. The arm will be used to press the buttons to start the sonicator.



Figure 5.4: Image of the grid of holes to repeatably hold tubes in the sonicator.

5.5 Spectrophotometer Integration

The only hardware change required for the spectrophotometer's integration into the robotic system is 3D-printed brackets for the feet similar to other equipment. The small lid can be opened and closed by the robotic arm, which is also how the machine's buttons can be pressed. However, There is currently no concrete plan to prepare the sample.

Chapter 6

Experiments, Testing, and Results

The results of this project are divided between full ATPE process experiments with partial automation, referred to as "hybrid ATPE", and a summary of the current status of each component of the automation system. The former is presented in Section 6.1, while the latter is given in Section 6.2.

6.1 Hybrid ATPE

Until the external robotic arm is integrated, it is not possible to test the ATPE automation system fully autonomously. However, much validation has been performed of the system in a semi-autonomous mode. This mode, referred to as "hybrid ATPE", requires human intervention to move tubes between the *OT-2* and centrifuge. Furthermore, this mode only allows for the main ATPE process, not the pre- or post-processing steps. Without the robotic arm, tubes must be manually moved to the UV-Vis spectrophotometer and sonicator. Additionally, a reliable powder dispenser for the automation of stock preparation, one of the pre-processing steps, is still lacking (Section 4.4).

6.1.1 Statement of Contributions

The testing of hybrid ATPE was performed by HeeBong Yang, Luke Neal, Matt Piatt, Nicki Caswell, Garen Simpson, and the author.

6.1.2 Hybrid ATPE Results

Hybrid ATPE has been performed at least 15 times, each instance consisting of multiple (~ 12) separations. Being more repeatable and efficient than manual ATPE, this hybrid method has significant advantages over its manual counterpart despite the absence of full automation.

The semi-autonomous mode is much more consistent than manual ATPE, which can be attributed to more repeatable pipetting by the *OT-2*. It was possible to partially recreate the SDS concentration curve from Fagan [68] using the hybrid ATPE method, which is presented in Figure 6.1b. After adjusting the fitting curve for the updated DOC concentration of 0.04 % [68], it is possible to notice a similar trend between the established results of Figure 6.1a and the results recreated by hybrid ATPE of Figure 6.1b. One notable exception to this is the chirality $(6, 6)$, which appears well above the curve for the hybrid ATPE method, despite being located slightly below the curve in the result of Fagan. Although the curve recreated with hybrid ATPE is not as complete as the established result, it is more exhaustive and matches more closely than the attempt to recreate the curve with manual ATPE, whose results are presented in Figure 6.1c. Figure 6.1a shows that an exhaustive curve is possible via manual ATPE, however this requires a significant investment of time and delicate micro-pipetting. Figure 6.1b and Figure 6.1c compare the results that can be obtained within a similar timespan (excluding the time required to develop the automated system) by a competent researcher.

This experiment used a "bottom extraction" ATPE recipe, meaning that the top phase always became a purified sample and the bottom phase became the new parent. Two similar but different recipes were used: one "collapsing" T and BT into a parent (noted X in Table 6.1), and the other using 2B directly. Using this method, 20 pure SWCNT samples were created after 24 separations. The samples have multiple chiralities, but a single chirality is dominant. The spectra for the samples produced during one such bottom extraction recipe are presented in Figure 6.2, where it is possible to see the dominant chirality of each sample.

Hybrid ATPE can also be more efficient than performing the procedure entirely manually. Qualitatively, some operators report that hybrid ATPE greatly improves efficiency. Indeed, it is possible to make preparations for the next step while the automated system is processing the current step. The elapsed times for each step of a bottom extraction recipe are shown in Table 6.1. An experienced operator is able to perform these steps in a similar amount of time. However, the automated system can be run unattended, so the operator can perform tasks manually while the system is running. Each sample must undergo centrifugation for 3 minutes to separate the phases, and each purified top phase

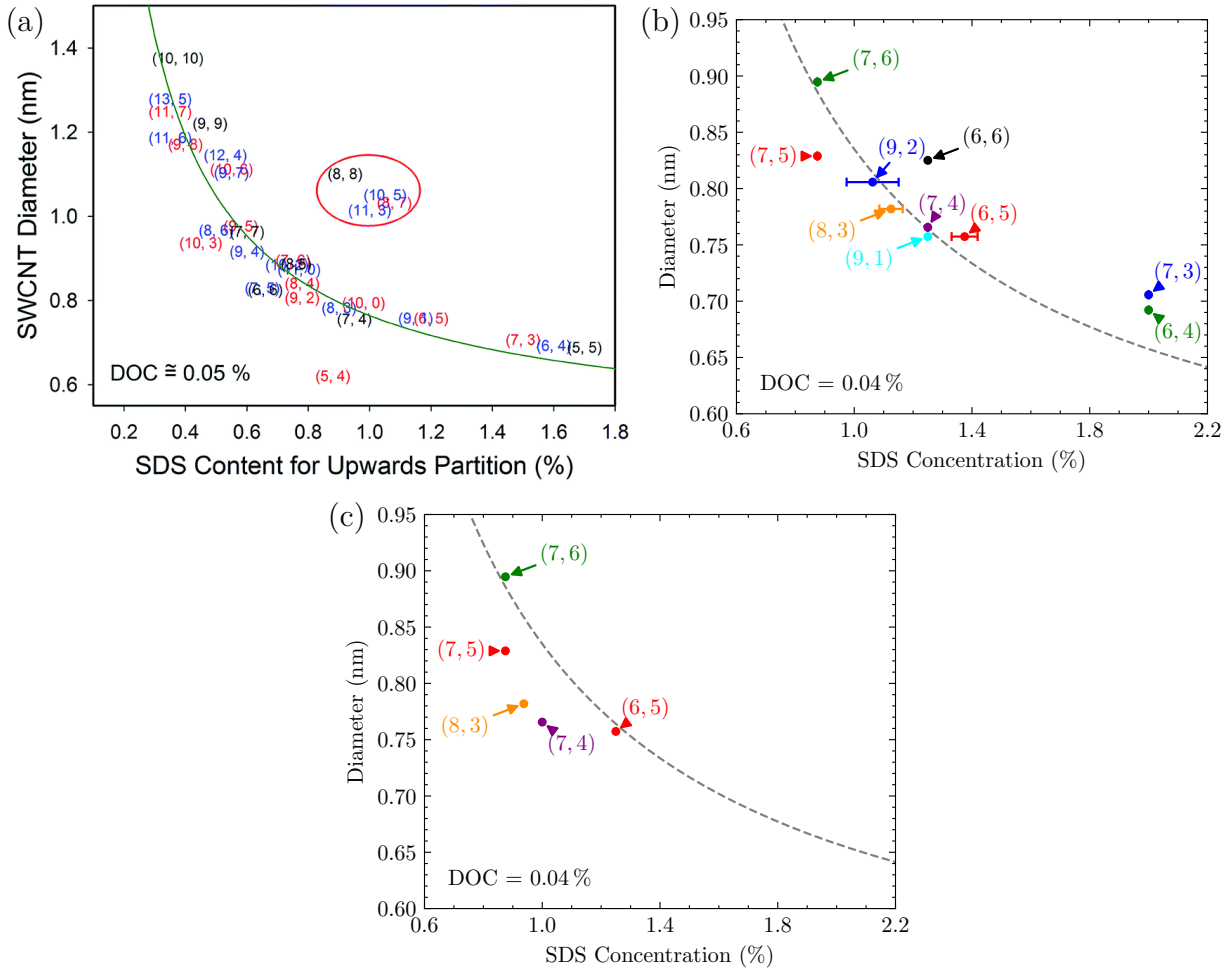


Figure 6.1: Experimental data showing the approximate SDS concentration required to partition the SWCNT with the annotated chirality into the top phase of a PEG/DX aqueous two-phase system with a fixed DX concentration (indicated in each plot). (a) Established results reproduced unmodified from Fagan [68] under the CC BY 3.0 license. The green curve is fit to the data and has the equation $D/\text{nm} = 0.284/(\text{SDS}/\%) + 0.48$ [67]. (b-c) Recreated results with hybrid ATPE (b) and manual ATPE (c) data from HeeBong Yang. The same fitting curve is used, except that the horizontal axis has been scaled by 4/5 to account for the difference in DX concentration. The colours have no significance and are used merely to distinguish chiralities. The error bars show the standard deviation between multiple subpopulations with the same chirality.

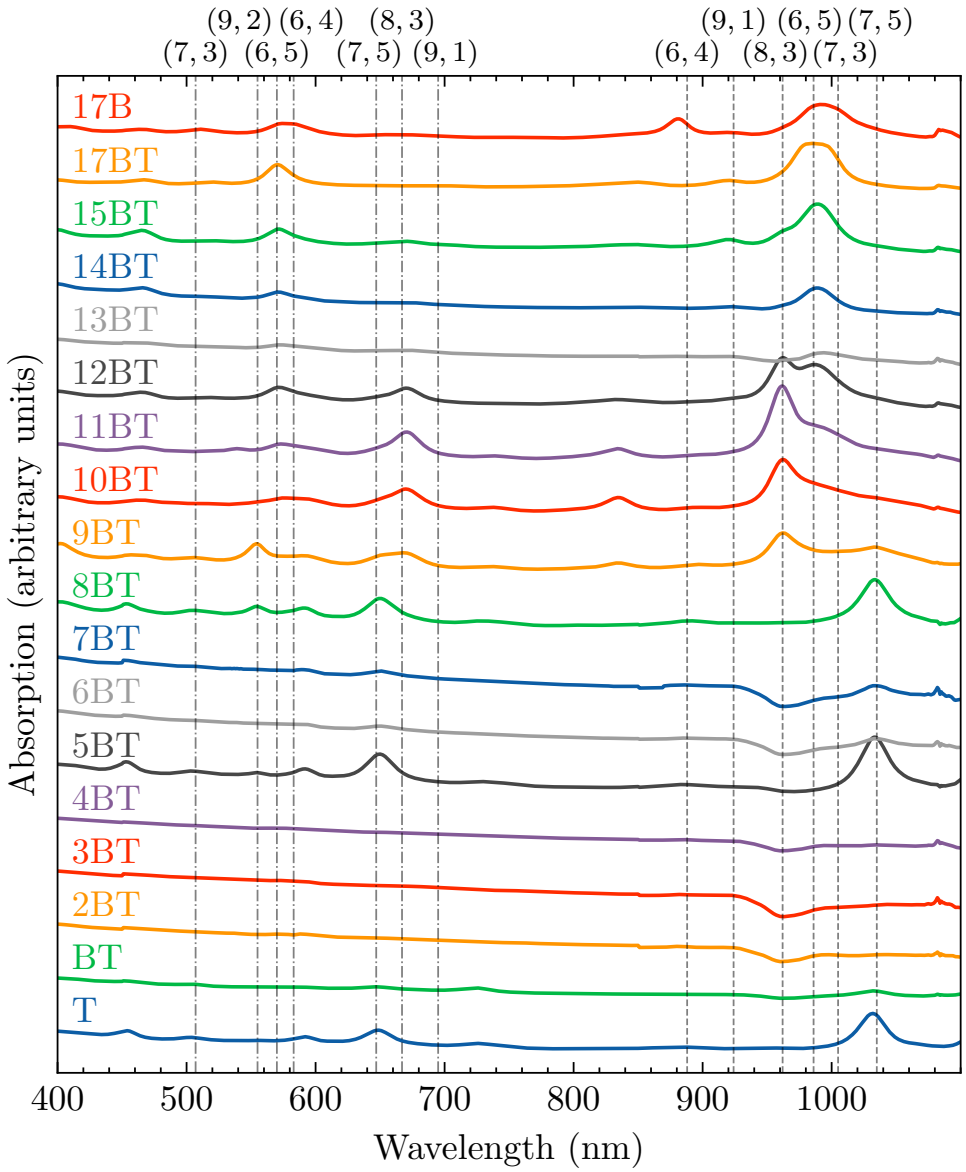


Figure 6.2: Spectrophotometer spectra of purified SWCNT samples produced by hybrid ATPE using a bottom-extraction recipe. Each spectrum is normalised at 570 nm and is given an arbitrary offset to separate it visually. The sample identifier is annotated in the same colour as the associated spectrum. The colours are arbitrary. The data are from HeeBong Yang. The vertical lines show the experimental wavelengths corresponding to S11 (dashed) and S22 (dash-dotted) band gap energies for important chiralities [178].

must be prepared for and analysed with the spectrophotometer. The former task cannot be run at the same time as the automated system for such a bottom extraction recipe, but can nearly double throughput for alternative recipes where both the top and bottom phases are used for subsequent separations, or for two parallel bottom extraction recipes.

Table 6.1: Time required by the *OT-2* for each step of a bottom extraction recipe. Data from HeeBong Yang

Operation	**Time (h:mm:ss)**
B	0:09:07
X (= T + BT)	0:04:44
XB	0:08:03
X2B	0:05:03
X3B	0:08:06
X4B	0:03:47
X5B	0:03:12
X6B	0:07:24
X7B	0:03:16
X8B	0:03:13
X9B	0:02:18
X10B	0:02:51
Total	1:01:04

6.2 Status of Each Component

Although the status of each module is discussed in its respective section, a summary of the system detailing which components are working and which are not is given here. Table 6.2 acts as a colour-coded index for the status of all subsystems, referencing the relevant section for each. This table shows that 8 modules (66.7 %) are complete, 2 (16.7 %) are works in progress, and 2 (16.7 %) are not working at all. The two that are not working at all are the *Powder Dispenser* and the external robotic arm. A working version of the former has been attempted five times, yet still has no satisfactory implementation. The latter is a recently-added expansion to the project that is scheduled as a second generation. A detailed status of all modules from Table 6.2 is given in the following paragraphs.

Table 6.2: Status of each module of the automation system. The green status indicates a fully working module, requiring no or only minor improvements. In most cases, these modules are actively being used in hybrid ATPE. The orange status indicates modules that have significant progress, but are not complete. The red status indicates that all attempts to implement a module have failed, or that the module is only in preliminary phases.

Module	**Relevant Section**	**Status**	**Remark**
OT-2	3.2.2	Working	
Gripper	4.1	Working	
Vial Opener	4.2	Working	Would benefit from a shorter design.
Separation	4.3		
Photo Stage	4.3.1	Working	
Computer Vision	4.3.2	Working	Would benefit from more testing.
Powder Handling System	4.4		
Powder Dispenser	4.4.2	Failed	Many designs have been attempted.
Weigh Module	4.4.4	Working	
Conveyor	4.4.3	Working	
Inventory System	4.5		
Barcode Scanner	4.5.2	Ongoing	No camera has been selected.
Barcode Printer	4.5.3	Ongoing	Missing written procedure on applying code.
Tracking Code	4.5.4	Working	
Robotic Arm and External Instruments	5	Little Progress	Scheduled for the next generation of the project.

Although the custom modules attached to it are subject to change, the *OT-2* itself is a complete off-the-shelf machine. Being the liquid handling robot around which the automation system is built, it is being used extensively in hybrid ATPE. The machine has limitations, such as being unable to transport vials and only being able to pipette vertically, but these are solved by custom modules. The only remaining grievances with this machine are in its software. However, being only software changes, these could be solved by future versions of the code. Some of these are already planned for the next major version of the software. Others are obviously bugs, which *Opentrons* should have an obligation to fix. Finally, since the software is open source, it would be possible to modify the code in order to implement application-specific features. In fact, this has already been done twice for this project.

The *Gripper* module works well and is being used in hybrid ATPE to transport samples from the vial rack into the *Photo Stage* and to remove the caps from them with the assistance of the *Vial Opener*. The module is well-tested and refined. Indeed, many of its parts have undergone multiple revisions, most notably the pincers. Some areas for improvement are the module's slow speed for both axes and bulky head.

The *Vial Opener* module has been tested moderately, but has never successfully been used in hybrid ATPE. The reason for this is that most executions of hybrid ATPE occurred before the *Vial Opener* cap storage system was finalised. In the most recent execution, there was an attempt to use the module. However, several problems were encountered, many of which being caused by the *OT-2* itself. Thus, the *Vial Opener* was never successfully used in hybrid ATPE. Still, the module is complete and no problems are foreseen for the next attempt once the *OT-2* bugs are resolved. The major flaw of this module is its height. The cup of the *Vial Opener* is so high that the brim of the 50 mL tube and 14.7 mL vial used sit above the maximum height of any pipette. This prevents liquid from being transferred to any container while it is inside the *Vial Opener* and causes problems for the calibration. The former is not a big problem given that the container is typically moved back to the vial rack once uncapped. This makes liquid transfers more efficient given that the source is either another vial from the same rack or a stock from the tube rack in an adjacent slot of the *OT-2* deck. The latter problem is more challenging to resolve. The *OT-2* performs a "labware calibration" before the run of every protocol. The tip of one of the machine's pipettes is moved to a container of each occupied slot of the deck, and the user is prompted to manually move the tip in 3-dimensional space with on-screen controls such that the tip is aligned with the centre of the container and that the end of the tip is touching the plane of the brim of the container. Since any container put inside the *Vial Opener* is higher than both pipettes of the *OT-2*, this calibration fails. A workaround was devised where a false container was added to the *Vial Opener*'s labware definition such that the calibration

could be performed with some reference marker other than the container in the cup. Still, a future revision of this module should minimize its height.

The *Photo Stage* of the automated separation subsystem works well and is being used in hybrid ATPE. The module is extremely simple, having only a single servo motor and a single relay to toggle the backlight. The other component of the separation system, the boundary detection computer vision code, is also being used in hybrid ATPE. This code works well and has successfully been used to automatically extract the top phase of a 2-phase sample using the *OT-2*. However, there are still some edge cases that are not well detected by the system. Most notably, the boundary is not properly detected when both phases are nearly totally transparent and the same colour. Until recently, the system did not work well when both phases were opaque and of similar colours. However, this was fixed by the inclusion of a second boundary detection method. This is explained in Section 4.3.2.6.

As for the *Powder Handling System*, most subcomponents are working well, specifically the *Conveyor* and the *Weigh Module*. Yet, a satisfactory *Powder Dispenser* is still lacking, as elaborated in Section 4.4.2. Five attempts have been made, but they all face the same problem: the powder, being extremely fine, works its way between sliding surfaces and binds the system. A prototype dispenser has been made without sliding surfaces, but this has not yet been refined into a complete module (Section 4.4.2.3). Since the dispenser is still lacking, the other components of the *Powder Handling System* have only been tested in isolation without ever being used properly. Furthermore, the *Powder Handling System* is for stock-making, which is not part of the main ATPE process.

The *Barcode Printer* and *Barcode Scanner* of the *Inventory System* are currently works in progress. A label maker has been used to print QR codes, but this process is done manually without a proper written procedure. The code size and location on the vial have yet to be standardised. Furthermore, although the automatic application of the codes to the vials is not currently in the scope of this project, it would be possible to programmatically generate and print the codes. This would save time for the operator and reduce the risk of human error. Many cameras and code types have been explored for the *Barcode Scanner*. However, a suitable camera has yet to be found. The current plan is to mount a downwards-facing camera above the top piece of acrylic of the *OT-2*. Each container will have a QR code attached to the top of its cap and its side. It is important that both parts be identifiable if they are separated. The camera will take a photo of the code on the top of all containers in the *OT-2*, locating and identifying them. Unfortunately, the cameras tested so far have had insufficient resolution to properly detect all QR codes from this distance. Research has been done into higher resolution cameras and appropriate lenses, but nothing has been implemented as of yet. The final component

of the *Inventory System*, the code to track the contents of each container, is working and can be used in some capacity without the other components. If the barcode of a sample is properly registered in the protocol file, its contents will automatically be updated in the database. This has also been integrated into other components. For example, the total volume of the sample is used as a hint for the computer vision system of where to expect the boundary between the top phase and the air above. Currently when registering a sample with this code, the barcode and location must be given. Once the *Barcode Scanner* is functional, the location will be determined automatically. The *Inventory System* is not crucial to ATPE, so hybrid ATPE can go on without it. The vials can be tracked by location with careful housekeeping rather than by barcode.

The final subsystem is the robotic arm and external instruments (centrifuge, sonicator, spectrophotometer). Although the robotic arm has been installed and a custom tabletop has been made to repeatably locate all instruments, this subsystem is in its early phases. This is a recently-added expansion to the project that is scheduled as a second generation. For the main ATPE process, the only advantage is that vials are automatically taken to the centrifuge by the robotic arm. In hybrid ATPE, the protocol is paused so that the samples can manually be taken to and from the centrifuge. The sonicator is used for the stock-making pre-processing step, and the spectrophotometer is used for the characterisation post-processing step.

Chapter 7

Conclusion

This thesis describes the development of a robotics system with the goal of automating the sorting of carbon nanotubes using the ATPE method. The heart of the system is the affordable *OT-2* liquid handling robot, around which multiple custom modules were added, enabling additional functionality required for ATPE. These modules include a *Gripper* to transport vials and tubes inside the *OT-2*, a *Vial Opener* to remove the caps of vials and tubes, a *Photo Stage* to take repeatable images of the phase boundary and accompanying image processing code to locate the boundary in the photo, a *Powder Handling System* to create the chemical stocks required for ATPE, and an *Inventory System* to track the contents and history of each process vial. Furthermore, there has been development of a robotic arm system capable of integrating instruments external to the *OT-2*. The most important is the centrifuge, which must be used at each step of the main ATPE process. Other external instruments include the sonicator, used during stock-making, and the spectrophotometer, used for classification.

The automated system is able to perform the majority of the ATPE process, the only interruption being a pausing of the protocol so that a human operator can manually move vials to the centrifuge. Thus, the system is incomplete until the external robotic system can be achieved. Still, the steps in most need of automation have been implemented. Indeed, separation is the step requiring the most skill and training, as the top phase must be carefully extracted via micro-pipetting without contaminating the phase boundary. Furthermore, tedious tasks such as mixing chemicals to carry out each step of the ATPE recipe have been automated. The time-consuming tasks of the main ATPE process, and those inviting human error, have been robotised. Transporting vials to and from a centrifuge is easy and does not require much training.

However, this does not mean that all error has been eliminated. Human error has been replaced by robot error given that automated separation still leaves more waste than manual separation and the phase boundary is contaminated more frequently. The big limitation in this regard is that the pipettes of the *OT-2* can only be operated vertically. When separation is performed manually, the pipette is tipped and rotated around the inside of the vial to aspirate the liquid from the top phase that is being wicked up the walls of the vial by surface tension. This is currently not possible with the *OT-2*, although there are plans to implement this using the external robotic arm or another device.

Finally, the pre- and post-processing steps are not working overall. Many attempts have been made to implement a satisfactory *Powder Handling System*, but they have all failed. Furthermore, the post-processing step of characterisation has been generally undeveloped except for a place's being reserved on the robotics table for the spectrophotometer. The focus has been on the main ATPE process.

7.1 Practical Lessons

If the project were to be redone, it could be advantageous to focus on the main ATPE process at first. The preparation of stocks is not the most time-consuming task of ATPE, nor is it the one with the highest chance of human error. Yet, a large time investment was given to the *Powder Handling System* for the development of the conveyor and three failed designs for the *Powder Dispenser*. Allocating this time to perfecting the main process might have been more valuable. A state enabling hybrid ATPE could have been reached more quickly, allowing more time for testing. In the intervening time, the *Powder Handling System* could have been tackled.

Every attempt to design a working *Powder Dispenser* had the same issue. They all used sliding surfaces with a tight tolerance to constrain the powder. However, the ultra-fine powder found its way between these surfaces and ultimately bound the system. Alternative designs, especially those without sliding surfaces, should be pursued. Furthermore, commercial products should be investigated.

7.2 Future Prospects

The external robotic arm system should be finalised so that the centrifuge, sonicator, and spectrophotometer can be integrated. The robotic arm has been mounted to the table and

tested. The biggest hurdle to overcome is enabling communication between the *OT-2* and the robotic arm. Once this is achieved, the arm can be programmed to bring vials to the centrifuge, to bring tubes to the sonicator, and to bring samples to the spectrophotometer.

A satisfactory *Powder Dispenser* is sorely lacking. This would finalise the *Powder Handling System* and enable stock-making inside the *OT-2*. Alternative designs and commercial products should be considered.

Bibliography

[1] Gage Hills, Christian Lau, Andrew Wright, *et al.*, "Modern microprocessor built from complementary carbon nanotube transistors," *Nature*, vol. 572, no. 7771, pp. 595–602, Aug. 2019. DOI: 10.1038/s41586-019-1493-8. [Online]. Available: https://doi.org/10.1038%2Fs41586-019-1493-8.

[2] Max M. Shulaker, Gage Hills, Nishant Patil, *et al.*, "Carbon nanotube computer," *Nature*, vol. 501, no. 7468, pp. 526–530, Sep. 2013. DOI: 10.1038/nature12502. [Online]. Available: https://doi.org/10.1038%2Fnature12502.

[3] Franz Kreupl, "Carbon-nanotube computer scaled up," *Nature*, vol. 572, no. 7771, pp. 588–589, Aug. 2019. DOI: 10.1038/d41586-019-02519-2. [Online]. Available: https://doi.org/10.1038%2Fd41586-019-02519-2.

[4] T. Maniecki, O. Shtyka, P. Mierczynski, *et al.*, "Carbon nanotubes: Properties, synthesis, and application," *Fibre Chemistry*, vol. 50, no. 4, pp. 297–300, Nov. 2018. DOI: 10.1007/s10692-019-09979-2. [Online]. Available: https://doi.org/10.1007%2Fs10692-019-09979-2.

[5] E. Amram Bengio, Damir Senic, Lauren W. Taylor, *et al.*, "High efficiency carbon nanotube thread antennas," *Applied Physics Letters*, vol. 111, no. 16, p. 163 109, Oct. 2017. DOI: 10.1063/1.4991822. [Online]. Available: https://doi.org/10.1063%2F1.4991822.

[6] W. Hoenlein, F. Kreupl, G.S. Duesberg, *et al.*, "Carbon nanotube applications in microelectronics," *IEEE Transactions on Components and Packaging Technologies*, vol. 27, no. 4, pp. 629–634, Dec. 2004. DOI: 10.1109/tcapt.2004.838876. [Online]. Available: https://doi.org/10.1109%2Ftcapt.2004.838876.

[7] XiaoJing Wang and Zhuang Liu, "Carbon nanotubes in biology and medicine: An overview," *Chinese Science Bulletin*, vol. 57, no. 2-3, pp. 167–180, Nov. 2011. DOI: 10.1007/s11434-011-4845-9. [Online]. Available: https://doi.org/10.1007%2Fs11434-011-4845-9.

[8] Eugenii Katz and Itamar Willner, "Biomolecule-functionalized carbon nanotubes: Applications in nanobioelectronics," *ChemPhysChem*, vol. 5, no. 8, pp. 1084–1104, Aug. 2004. DOI: 10.1002/cphc.200400193. [Online]. Available: https://doi.org/10.1002%2Fcphc.200400193.

[9] Sobhi Daniel, Talasila Prasada Rao, Kota Sreenivasa Rao, *et al.*, "A review of DNA functionalized/grafted carbon nanotubes and their characterization," *Sensors and Actuators B: Chemical*, vol. 122, no. 2, pp. 672–682, Mar. 2007. DOI: 10.1016/j.snb.2006.06.014. [Online]. Available: https://doi.org/10.1016%2Fj.snb.2006.06.014.

[10] Daniel A. Heller, Esther S. Jeng, T. K. Yeung, *et al.*, "Optical detection of dna conformational polymorphism on single-walled carbon nanotubes," *Science*, vol. 311, pp. 508–511, 2006.

[11] Zhuang Liu, Xiaolin Li, Scott M. Tabakman, Kaili Jiang, Shoushan Fan, and Hongjie Dai, "Multiplexed multicolor raman imaging of live cells with isotopically modified single walled carbon nanotubes," *Journal of the American Chemical Society*, vol. 130, no. 41, pp. 13 540–13 541, Sep. 2008. DOI: 10.1021/ja806242t. [Online]. Available: https://doi.org/10.1021%2Fja806242t.

[12] Kevin Welsher, Zhuang Liu, Dan Daranciang, and H. Dai, "Selective probing and imaging of cells with single walled carbon nanotubes as near-infrared fluorescent molecules.," *Nano letters*, vol. 8 2, pp. 586–90, 2008.

[13] Hui-Ming Cheng, Quan-Hong Yang, and Chang Liu, "Hydrogen storage in carbon nanotubes," *Carbon*, vol. 39, no. 10, pp. 1447–1454, Aug. 2001. DOI: 10.1016/s0008-6223(00)00306-7. [Online]. Available: https://doi.org/10.1016%2Fs0008-6223%2800%2900306-7.

[14] A. C. Dillon, K. M. Jones, T. A. Bekkedahl, C. H. Kiang, D. S. Bethune, and M. J. Heben, "Storage of hydrogen in single-walled carbon nanotubes," *Nature*, vol. 386, no. 6623, pp. 377–379, Mar. 1997. DOI: 10.1038/386377a0. [Online]. Available: https://doi.org/10.1038%2F386377a0.

[15] Jian Ping Lu, "Elastic properties of carbon nanotubes and nanoropes," *Physical Review Letters*, vol. 79, no. 7, pp. 1297–1300, Aug. 1997. DOI: 10.1103/physrevlett.79.1297. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.79.1297.

[16] A. Krishnan, E. Dujardin, T. W. Ebbesen, P. N. Yianilos, and M. M. J. Treacy, "Young's modulus of single-walled nanotubes," *Physical Review B*, vol. 58, no. 20, pp. 14 013–14 019, Nov. 1998. DOI: 10.1103/physrevb.58.14013. [Online]. Available: https://doi.org/10.1103%2Fphysrevb.58.14013.

[17] Thien An Le, Min Sik Kim, Sae Ha Lee, Tae Wook Kim, and Eun Duck Park, "CO and CO 2 methanation over supported ni catalysts," *Catalysis Today*, vol. 293-294, pp. 89–96, Sep. 2017. DOI: 10.1016/j.cattod.2016.12.036. [Online]. Available: https://doi.org/10.1016%2Fj.cattod.2016.12.036.

[18] S.A. Chernyak, E.V. Suslova, A.S. Ivanov, *et al.*, "Co catalysts supported on oxidized CNTs: Evolution of structure during preparation, reduction and catalytic test in fischer-tropsch synthesis," *Applied Catalysis A: General*, vol. 523, pp. 221–229, Aug. 2016. DOI: 10.1016/j.apcata.2016.06.012. [Online]. Available: https://doi.org/10.1016%2Fj.apcata.2016.06.012.

[19] Ming-Chi Tsai, Tsung-Kuang Yeh, and Chuen-Horng Tsai, "An improved electrodeposition technique for preparing platinum and platinum–ruthenium nanoparticles on carbon nanotubes directly grown on carbon cloth for methanol oxidation," *Electrochemistry Communications*, vol. 8, no. 9, pp. 1445–1452, Sep. 2006. DOI: 10.1016/j.elecom.2006.07.003. [Online]. Available: https://doi.org/10.1016%2Fj.elecom.2006.07.003.

[20] Sumio Iijima, "Helical microtubules of graphitic carbon," *Nature*, vol. 354, no. 6348, pp. 56–58, Nov. 1991. DOI: 10.1038/354056a0. [Online]. Available: https://doi.org/10.1038%2F354056a0.

[21] Sumio Iijima and Toshinari Ichihashi, "Single-shell carbon nanotubes of 1-nm diameter," *Nature*, vol. 363, no. 6430, pp. 603–605, Jun. 1993. DOI: 10.1038/363603a0. [Online]. Available: https://doi.org/10.1038%2F363603a0.

[22] LV Radushkevich and V áM Lukyanovich, "O strukture ugleroda, obrazujucegosja pri termiceskom razlozenii okisi ugleroda na zeleznom kontakte," *Zurn Fisic Chim*, vol. 26, no. 1, pp. 88–95, 1952.

[23] Marc Monthioux and Vladimir L. Kuznetsov, "Who should be given the credit for the discovery of carbon nanotubes?" *Carbon*, vol. 44, no. 9, pp. 1621–1623, Aug. 2006. DOI: 10.1016/j.carbon.2006.03.019. [Online]. Available: https://doi.org/10.1016%2Fj.carbon.2006.03.019.

[24] D. S. Bethune, C. H. Kiang, M. S. de Vries, *et al.*, "Cobalt-catalysed growth of carbon nanotubes with single-atomic-layer walls," *Nature*, vol. 363, no. 6430, pp. 605–607, Jun. 1993. DOI: 10.1038/363605a0. [Online]. Available: https://doi.org/10.1038%2F363605a0.

[25] G Dresselhaus, Mildred S Dresselhaus, and Riichiro Saito, *Physical properties of carbon nanotubes*. World scientific, 1998.

[26] Ming Zheng, "Sorting carbon nanotubes," *Topics in Current Chemistry*, vol. 375, no. 1, Jan. 2017. DOI: 10.1007/s41061-016-0098-z. [Online]. Available: https://doi.org/10.1007/2Fs41061-016-0098-z.

[27] Ralph Krupke, Frank Hennrich, Hilbert v. Löhneysen, and Manfred M. Kappes, "Separation of metallic from semiconducting single-walled carbon nanotubes," *Science*, vol. 301, no. 5631, pp. 344–347, Jul. 2003. DOI: 10.1126/science.1086534. [Online]. Available: https://doi.org/10.1126%2Fscience.1086534.

[28] F. Villalpando-Paez, H. Son, D. Nezich, *et al.*, "Raman spectroscopy study of isolated double-walled carbon nanotubes with different metallic and semiconducting configurations," *Nano Letters*, vol. 8, no. 11, pp. 3879–3886, Oct. 2008. DOI: 10.1021/nl802306t. [Online]. Available: https://doi.org/10.1021%2Fnl802306t.

[29] Young-Gui Yoon, Paul Delaney, and Steven G. Louie, "Quantum conductance of multiwall carbon nanotubes," *Physical Review B*, vol. 66, no. 7, Aug. 2002. DOI: 10.1103/physrevb.66.073407. [Online]. Available: https://doi.org/10.1103%2Fphysrevb.66.073407.

[30] HeeBong Yang, "Towards scalable electronic devices with sorted carbon nanotube thin-films," M.S. thesis, University of Waterloo, Jan. 2020, p. 135.

[31] Geyou Ao, Jason K. Streit, Jeffrey A. Fagan, and Ming Zheng, "Differentiating left- and right-handed carbon nanotubes by DNA," *Journal of the American Chemical Society*, vol. 138, no. 51, pp. 16 677–16 685, Dec. 2016. DOI: 10.1021/jacs.6b09135. [Online]. Available: https://doi.org/10.1021%2Fjacs.6b09135.

[32] Neha Arora and N.N. Sharma, "Arc discharge synthesis of carbon nanotubes: Comprehensive review," *Diamond and Related Materials*, vol. 50, pp. 135–150, Nov. 2014. DOI: 10.1016/j.diamond.2014.10.001. [Online]. Available: https://doi.org/10.1016%2Fj.diamond.2014.10.001.

[33] Roger Bacon, "Growth, structure, and properties of graphite whiskers," *Journal of Applied Physics*, vol. 31, no. 2, pp. 283–290, Feb. 1960. DOI: 10.1063/1.1735559. [Online]. Available: https://doi.org/10.1063%2F1.1735559.

[34] W. Krätschmer, Lowell D. Lamb, K. Fostiropoulos, and Donald R. Huffman, "Solid c60: A new form of carbon," *Nature*, vol. 347, no. 6291, pp. 354–358, Sep. 1990. DOI: 10.1038/347354a0. [Online]. Available: https://doi.org/10.1038%2F347354a0.

[35] Jan Prášek, Jana Drbohlavová, Jana Chomoucká, *et al.*, "Methods for carbon nanotubes synthesis—review," eng, *Journal of Materials Chemistry*, vol. 21, 2011, ISSN: 0959-9428. DOI: http://dx.doi.org/10.1039/c1jm12254a. [Online]. Available: http://pubs.rsc.org/en/content/articlelanding/2011/jm/c1jm12254a.

[36] T. Guo, P. Nikolaev, A. Thess, D.T. Colbert, and R.E. Smalley, "Catalytic growth of single-walled manotubes by laser vaporization," *Chemical Physics Letters*, vol. 243, no. 1-2, pp. 49–54, Sep. 1995. DOI: 10.1016/0009-2614(95)00825-o. [Online]. Available: https://doi.org/10.1016%2F0009-2614%2895%2900825-o.

[37] Rasel Das, Zohreh Shahnavaz, Md. Eaqub Ali, Mohammed Moinul Islam, and Sharifah Bee Abd Hamid, "Can we optimize arc discharge and laser ablation for well-controlled carbon nanotube synthesis?" *Nanoscale Research Letters*, vol. 11, no. 1, Nov. 2016. DOI: 10.1186/s11671-016-1730-0. [Online]. Available: https://doi.org/10.1186%2Fs11671-016-1730-0.

[38] Zan Lu, Raad Raad, Farzad Safaei, Jiangtao Xi, Zhoufeng Liu, and Javad Foroughi, "Carbon nanotube based fiber supercapacitor as wearable energy storage," *Frontiers in Materials*, vol. 6, Jun. 2019. DOI: 10.3389/fmats.2019.00138. [Online]. Available: https://doi.org/10.3389%2Ffmats.2019.00138.

[39] Mukul Kumar and Yoshinori Ando, "Gigas growth of carbon nanotubes," *Defence Science Journal*, vol. 58, no. 4, pp. 496–503, Jul. 2008. DOI: 10.14429/dsj.58.1670. [Online]. Available: https://doi.org/10.14429%2Fdsj.58.1670.

[40] W. Z. Li, S. S. Xie, L. X. Qian, *et al.*, "Large-scale synthesis of aligned carbon nanotubes," *Science*, vol. 274, no. 5293, pp. 1701–1703, Dec. 1996. DOI: 10.1126/science.274.5293.1701. [Online]. Available: https://doi.org/10.1126%2Fscience.274.5293.1701.

[41] Sivakumar VM, Abdul Rahman Mohamed, Ahmad Zuhairi Abdullah, and Siang-Piao Chai, "Role of reaction and factors of carbon nanotubes growth in chemical vapour decomposition process using methane—a highlight," *Journal of Nanomaterials*, vol. 2010, pp. 1–11, 2010. DOI: 10.1155/2010/395191. [Online]. Available: https://doi.org/10.1155/2010/395191.

[42] Olga Zaytseva and Günter Neumann, "Carbon nanomaterials: Production, impact on plant development, agricultural and environmental applications," *Chemical and Biological Technologies in Agriculture*, vol. 3, no. 1, Jul. 2016. DOI: 10.1186/s40538-016-0070-8. [Online]. Available: https://doi.org/10.1186%2Fs40538-016-0070-8.

[43] Jan Prasek, Jana Drbohlavova, Jana Chomoucka, *et al.*, "Methods for carbon nanotubes synthesis—review," *Journal of Materials Chemistry*, vol. 21, no. 40, p. 15 872, 2011. DOI: 10.1039/c1jm12254a. [Online]. Available: https://doi.org/10.1039/c1jm12254a.

[44] Pavel Nikolaev, Michael J Bronikowski, R.Kelley Bradley, *et al.*, "Gas-phase catalytic growth of single-walled carbon nanotubes from carbon monoxide," *Chemical Physics Letters*, vol. 313, no. 1-2, pp. 91–97, Nov. 1999. DOI: 10.1016/s0009-2614(99)01029-5. [Online]. Available: https://doi.org/10.1016%2Fs0009-2614%2899%2901029-5.

[45] Michael J. Bronikowski, Peter A. Willis, Daniel T. Colbert, K. A. Smith, and Richard E. Smalley, "Gas-phase production of carbon single-walled nanotubes from carbon monoxide via the HiPco process: A parametric study," *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, vol. 19, no. 4, pp. 1800–1805, Jul. 2001. DOI: 10.1116/1.1380721. [Online]. Available: https://doi.org/10.1116%2F1.1380721.

[46] D.E. Resasco, W.E. Alvarez, F. Pompeo, *et al.*, *Journal of Nanoparticle Research*, vol. 4, no. 1/2, pp. 131–136, 2002. DOI: 10.1023/a:1020174126542. [Online]. Available: https://doi.org/10.1023%2Fa%3A1020174126542.

[47] Ming Zheng, Anand Jagota, Michael S. Strano, *et al.*, "Structure-based carbon nanotube sorting by sequence-dependent DNA assembly," *Science*, vol. 302, no. 5650, pp. 1545–1548, Nov. 2003. DOI: 10.1126/science.1091911. [Online]. Available: https://doi.org/10.1126%2Fscience.1091911.

[48] Ming Zheng, Anand Jagota, Ellen D. Semke, *et al.*, "DNA-assisted dispersion and separation of carbon nanotubes," *Nature Materials*, vol. 2, no. 5, pp. 338–342, Apr. 2003. DOI: 10.1038/nmat877. [Online]. Available: https://doi.org/10.1038%2Fnmat877.

[49] Michael S. Arnold, Samuel I. Stupp, and Mark C. Hersam, "Enrichment of single-walled carbon nanotubes by diameter in density gradients," *Nano Letters*, vol. 5, no. 4, pp. 713–718, Mar. 2005. DOI: 10.1021/nl050133o. [Online]. Available: https://doi.org/10.1021%2Fnl050133o.

[50] W. Wenseleers, I. I. Vlasov, E. Goovaerts, E. D. Obraztsova, A. S. Lobach, and A. Bouwen, "Efficient isolation and solubilization of pristine single-walled nanotubes in bile salt micelles," *Advanced Functional Materials*, vol. 14, no. 11, pp. 1105–1112, Nov. 2004. DOI: 10.1002/adfm.200400130. [Online]. Available: https://doi.org/10.1002%2Fadfm.200400130.

[51] Saunab Ghosh, Sergei M. Bachilo, and R. Bruce Weisman, "Advanced sorting of single-walled carbon nanotubes by nonlinear density-gradient ultracentrifugation," *Nature Nanotechnology*, vol. 5, no. 6, pp. 443–450, May 2010. DOI: 10.1038/nnano.2010.68. [Online]. Available: https://doi.org/10.1038%2Fnnano.2010.68.

[52] Adrian Nish, Jeong-Yuan Hwang, James Doig, and Robin J. Nicholas, "Highly selective dispersion of single-walled carbon nanotubes using aromatic polymers," *Nature Nanotechnology*, vol. 2, no. 10, pp. 640–646, Sep. 2007. DOI: 10.1038/nnano.2007.290. [Online]. Available: https://doi.org/10.1038%2Fnnano.2007.290.

[53] Kevin S. Mistry, Brian A. Larsen, and Jeffrey L. Blackburn, "High-yield dispersions of large-diameter semiconducting single-walled carbon nanotubes with tunable narrow chirality distributions," *ACS Nano*, vol. 7, no. 3, pp. 2231–2239, Feb. 2013. DOI: 10.1021/nn305336x. [Online]. Available: https://doi.org/10.1021%2Fnn305336x.

[54] Yuichi Kato, Masashi Fukuzawa, Fumiyuki Toshimitsu, and Naotoshi Nakashima, "Separation of semiconducting single-walled carbon nanotubes using a flavin compound," *Chemistry Letters*, vol. 44, no. 4, pp. 566–567, Apr. 2015. DOI: 10.1246/cl.141193. [Online]. Available: https://doi.org/10.1246%2Fcl.141193.

[55] Jie Han, Qiyan Ji, Hongbo Li, *et al.*, "A photodegradable hexaaza-pentacene molecule for selective dispersion of large-diameter semiconducting carbon nanotubes," *Chemical Communications*, vol. 52, no. 49, pp. 7683–7686, 2016. DOI: 10.1039/c6cc01558a. [Online]. Available: https://doi.org/10.1039%2Fc6cc01558a.

[56] Takeshi Tanaka, Hehua Jin, Yasumitsu Miyata, *et al.*, "Simple and scalable gel-based separation of metallic and semiconducting carbon nanotubes," *Nano Letters*, vol. 9, no. 4, pp. 1497–1500, Feb. 2009. DOI: 10.1021/nl8034866. [Online]. Available: https://doi.org/10.1021%2Fnl8034866.

[57] Huaping Liu, Daisuke Nishide, Takeshi Tanaka, and Hiromichi Kataura, "Large-scale single-chirality separation of single-wall carbon nanotubes by simple gel chromatography," *Nature Communications*, vol. 2, no. 1, May 2011. DOI: 10.1038/ncomms1313. [Online]. Available: https://doi.org/10.1038%2Fncomms1313.

[58] António Lima Grilo, M. Raquel Aires-Barros, and Ana M. Azevedo, "Partitioning in aqueous two-phase systems: Fundamentals, applications and trends," *Separation & Purification Reviews*, vol. 45, no. 1, pp. 68–80, Nov. 2014. DOI: 10.1080/15422119.2014.983128. [Online]. Available: https://doi.org/10.1080%2F15422119.2014.983128.

[59] Per-Åke Albertsson, "Partition of cell particles and macromolecules in polymer two-phase systems," in *Advances in Protein Chemistry*, Elsevier, 1970, pp. 309–341. DOI: 10.1016/s0065-3233(08)60244-2. [Online]. Available: https://doi.org/10.1016%2Fs0065-3233%2808%2960244-2.

[60] Mujahid Iqbal, Yanfei Tao, Shuyu Xie, *et al.*, "Aqueous two-phase system (ATPS): An overview and advances in its applications," *Biological Procedures Online*, vol. 18, no. 1, Oct. 2016. DOI: 10.1186/s12575-016-0048-8. [Online]. Available: https://doi.org/10.1186%2Fs12575-016-0048-8.

[61] Constantine Y Khripin, Jeffrey A. Fagan, and Ming Zheng, "Spontaneous partition of carbon nanotubes in polymer-modified aqueous phases," *Journal of the American Chemical Society*, vol. 135, no. 18, pp. 6822–6825, Apr. 2013. DOI: 10.1021/ja402762e. [Online]. Available: https://doi.org/10.1021%2Fja402762e.

[62] Hui Gui, Jason K. Streit, Jeffrey A. Fagan, Angela R. Hight Walker, Chongwu Zhou, and Ming Zheng, "Redox sorting of carbon nanotubes," *Nano Letters*, vol. 15, no. 3, pp. 1642–1646, Mar. 2015. DOI: 10.1021/nl504189p. [Online]. Available: https://doi.org/10.1021%2Fnl504189p.

[63] Geyou Ao, Constantine Y. Khripin, and Ming Zheng, "DNA-controlled partition of carbon nanotubes in polymer aqueous two-phase systems," *Journal of the American Chemical Society*, vol. 136, no. 29, pp. 10 383–10 392, Jul. 2014. DOI: 10.1021/ja504078b. [Online]. Available: https://doi.org/10.1021%2Fja504078b.

[64] Jeffrey A. Fagan, Constantine Y. Khripin, Carlos A. Silvera Batista, *et al.*, "Isolation of specific small-diameter single-wall carbon nanotube species via aqueous two-phase extraction," *Advanced Materials*, vol. 26, no. 18, pp. 2800–2804, Jan. 2014. DOI: 10.1002/adma.201304873. [Online]. Available: https://doi.org/10.1002%2Fadma.201304873.

[65] Navaneetha K. Subbaiyan, Sofie Cambré, A. Nicholas G. Parra-Vasquez, Erik H. Hároz, Stephen K. Doorn, and Juan G. Duque, "Role of surfactants and salt in aqueous two-phase separation of carbon nanotubes toward simple chirality isolation," *ACS Nano*, vol. 8, no. 2, pp. 1619–1628, Jan. 2014. DOI: 10.1021/nn405934y. [Online]. Available: https://doi.org/10.1021%2Fnn405934y.

[66] Min Zhang, Constantine Y Khripin, Jeffrey A. Fagan, Peter McPhie, Yoichiro Ito, and Ming Zheng, "Single-step total fractionation of single-wall carbon nanotubes by countercurrent chromatography," *Analytical Chemistry*, vol. 86, no. 8, pp. 3980–3984, Apr. 2014. DOI: 10.1021/ac5003189. [Online]. Available: https://doi.org/10.1021%2Fac5003189.

[67] Jeffrey A. Fagan, Erik H. Hároz, Rachelle Ihly, *et al.*, "Isolation of >1 nm diameter single-wall carbon nanotube species using aqueous two-phase extraction," *ACS Nano*, vol. 9, no. 5, pp. 5377–5390, Apr. 2015. DOI: 10.1021/acsnano.5b01123. [Online]. Available: https://doi.org/10.1021%2Facsnano.5b01123.

[68] Jeffrey A. Fagan, "Aqueous two-polymer phase extraction of single-wall carbon nanotubes using surfactants," *Nanoscale Advances*, vol. 1, no. 9, pp. 3307–3324, 2019. DOI: 10.1039/c9na00280d. [Online]. Available: https://doi.org/10.1039%2Fc9na00280d.

[69] Michael S. Arnold, Alexander A. Green, James F. Hulvat, Samuel I. Stupp, and Mark C. Hersam, "Sorting carbon nanotubes by electronic structure using density differentiation," *Nature Nanotechnology*, vol. 1, no. 1, pp. 60–65, Oct. 2006. DOI: 10.1038/nnano.2006.52. [Online]. Available: https://doi.org/10.1038%2Fnnano.2006.52.

[70] Alexander A. Green, Matthew C. Duch, and Mark C. Hersam, "Isolation of single-walled carbon nanotube enantiomers by density differentiation," *Nano Research*, vol. 2, no. 1, pp. 69–77, Jan. 2009. DOI: 10.1007/s12274-009-9006-y. [Online]. Available: https://doi.org/10.1007%2Fs12274-009-9006-y.

[71] B.Yu. Zaslavsky, L.M. Miheeva, G.Z. Gasanova, and A.U. Mahmudov, "Effect of polymer composition on the relative hydrophobicity of the phases of the biphasic system aqueous dextran-poly(ethylene glycol)," *Journal of Chromatography A*, vol. 403, pp. 123–130, Jan. 1987. DOI: 10.1016/s0021-9673(00)96346-5. [Online]. Available: https://doi.org/10.1016%2Fs0021-9673%2800%2996346-5.

[72] Adam J. Blanch, Claire E. Lenehan, and Jamie S. Quinton, "Optimizing surfactant concentrations for dispersion of single-walled carbon nanotubes in aqueous solution," *The Journal of Physical Chemistry B*, vol. 114, no. 30, pp. 9805–9811, Jul. 2010. DOI: 10.1021/jp104113d. [Online]. Available: https://doi.org/10.1021%2Fjp104113d.

[73] NanoIntegris Inc. "Specification of purified swnts by hipco." (), [Online]. Available: https://nanointegris.com/our-products/small-diameter-swnts-hipco/ (visited on 06/03/2022).

[74] Sigma-Aldrich Inc. "Specification of mwnt." (), [Online]. Available: https://www.sigmaaldrich.com/catalog/product/aldrich/698849?lang=en&region=CA (visited on 06/03/2022).

[75] Sigma-Aldrich Inc. "Specification of sg65i." (), [Online]. Available: https://www.sigmaaldrich.com/catalog/product/aldrich/773735?lang=en&region=CA (visited on 06/03/2022).

[76] Sigma-Aldrich Inc. "Specification of sg76." (), [Online]. Available: https://www.sigmaaldrich.com/catalog/product/aldrich/704121?lang=en&region=CA (visited on 06/03/2022).

[77] Sigma-Aldrich Inc. "Specification of swnt." (), [Online]. Available: https://www.sigmaaldrich.com/catalog/product/aldrich/704113?lang=en&region=CA (visited on 06/03/2022).

[78] Beckman Coulter, Inc. "Optima™ XE-100." (2022), [Online]. Available: https://www.beckman.com/centrifuges/ultracentrifuges/optima-xe/a94516 (visited on 06/03/2022).

[79] Corning® Incorporated. "Corning® 50 mL Centrifuge Tubes." (2022), [Online]. Available: https://ecatalog.corning.com/life-sciences/b2c/US/en/Liquid-Handling/Tubes%2C-Liquid-Handling/Centrifuge-Tubes/Corning%C2%AE-50-mL-Centrifuge-Tubes/p/50mLCentrifugeTubes (visited on 06/08/2022).

[80] NanoIntegris Inc. "High purity, electronically separated nanomaterials." (2022), [Online]. Available: https://nanointegris.com/ (visited on 06/08/2022).

[81] Merck KGaA. "Milliporesigma | canada." (2022), [Online]. Available: https://www.sigmaaldrich.com/CA/en (visited on 06/08/2022).

[82] Jeffrey A. Fagan, "Electronic supporting information (ESI) for aqueous two-polymer phase extraction of single-wall carbon nanotubes using surfactants," *Nanoscale Advances*, vol. 1, no. 9, pp. 3307–3324, 2019. DOI: 10.1039/c9na00280d. [Online]. Available: https://doi.org/10.1039%2Fc9na00280d.

[83] Thermo Scientific™. "Sorvall™ ST 8 small benchtop centrifuge." (2022), [Online]. Available: https://www.thermofisher.com/order/catalog/product/75007200 (visited on 06/08/2022).

[84] VWR International, LLC. "VWR® Spectrophotometers, UV-Vis Scanning UV-3100PC and Vis V-3000-PC." (2022), [Online]. Available: https://us.vwr.com/store/product/12243112/vwr-spectrophotometers-uv-vis-scanning-uv-3100pc-and-vis-v-3000-pc (visited on 06/07/2022).

[85] Opentrons. "Open-source Lab Automation, starting at $5,000." (2022), [Online]. Available: https://opentrons.com/ (visited on 06/16/2022).

[86] Andrew Alliance. "Pipetting robot using electronic pipettes for liquid handling - Andrew+." (2022), [Online]. Available: https://www.andrewalliance.com/pipetting-robot/ (visited on 06/16/2022).

[87] Anton Paar GmbH. (2022), [Online]. Available: https://www.anton-paar.com/corp-en/products/details/sample-processor-automatic-dosing-blending-subsampling-transfering-forwarding-of-liquid-samples/ (visited on 06/16/2022).

[88] Sirius Automation Group Inc. "MultiTasker®." (2022), [Online]. Available: https://siriusautomation.com/multitasker/ (visited on 06/16/2022).

[89] Sirius Automation Group Inc. "MiniTasker®." (2022), [Online]. Available: https://siriusautomation.com/minitasker/ (visited on 06/16/2022).

[90] Sirius Automation Group Inc. "MicroTasker®." (2022), [Online]. Available: https://siriusautomation.com/microtasker/ (visited on 06/16/2022).

[91] Sirius Automation Group Inc. "OmniTasker®." (2022), [Online]. Available: https://siriusautomation.com/omnitasker/ (visited on 06/16/2022).

[92] "Opentrons/opentrons: Software for writing protocols and running them on the Opentrons OT-2." (2022), [Online]. Available: https://github.com/Opentrons/opentrons (visited on 05/10/2022).

[93] Opentrons. "OT2 Robot Hardware Files." (2018), [Online]. Available: https://github.com/Opentrons/ot2 (visited on 05/02/2022).

[94] Opentrons. "OT-2 Liquid Handler | Opentrons Lab Automation from $5,000." (2022), [Online]. Available: https://opentrons.com/ot-5 (visited on 06/16/2022).

[95] Blender Foundation. "blender.org - Home of the Blender project - Free and Open 3D Creation Software." (2022), [Online]. Available: https://www.blender.org/ (visited on 05/02/2022).

[96] ST. "STM32 Nucleo-32 development board with STM32F303K8 MCU, supports Arduino nano connectivity." (2022), [Online]. Available: https://www.st.com/en/evaluation-tools/nucleo-f303k8.html (visited on 05/02/2022).

[97] Trinamic. "TMC2209-LA." (2022), [Online]. Available: https://www.trinamic.com/products/integrated-circuits/details/tmc2209-la/ (visited on 05/02/2022).

[98] FTDI. "FT232RL." (2022), [Online]. Available: https://ftdichip.com/products/ft232rl/ (visited on 05/02/2022).

[99] Autodesk. "EAGLE | PCB Design And Electrical Schematic Software." (2022), [Online]. Available: https://www.autodesk.com/products/eagle/overview?term=1-YEAR&tab=subscription (visited on 05/02/2022).

[100] ams-OSRAM AG. "AS5600 Position Sensor." (2022), [Online]. Available: https://ams.com/en/AS5600 (visited on 05/02/2022).

[101] Colin Robertson, nxtn, Mike Jones, Gordon Hogenson, and Saisang Cai. "C enumeration declarations." (2021), [Online]. Available: https://docs.microsoft.com/en-us/cpp/c-language/c-enumeration-declarations?view=msvc-170 (visited on 05/05/2022).

[102] Arm Limited. "Free open source IoT OS and development tools from Arm." (2022), [Online]. Available: https://os.mbed.com/ (visited on 05/05/2022).

[103] National Electrical Manufacturers Association. "Motion/Position Control Motors, Controls and Feedback Devices." section 4.3.1.1. (2004), [Online]. Available: https://www.nema.org/Standards/view/Motion-Position-Control-Motors-Controls-and-Feedback-Devices (visited on 05/21/2022).

[104] RepRap. "NEMA motor." (2015), [Online]. Available: https://reprap.org/wiki/NEMA_Motor (visited on 05/21/2022).

[105] RepRap. "NEMA 17 stepper motor." (2020), [Online]. Available: https://reprap.org/wiki/NEMA_17_Stepper_motor (visited on 05/21/2022).

[106] YUJILEDS High CRI Webstore. "YUJILEDS® High CRI 95+ High Efficacy High Brightness 2835M LED Flexible Strip - 128 LEDs/m - Pack: 5m/reel." (2022), [Online]. Available: https://store.yujiintl.com/products/yujileds-high-cri-95-high-efficacy-2835m-led-flexible-strip (visited on 05/17/2022).

[107] Adrian Rosebrock. "Detecting ArUco markers with OpenCV and Python." (Dec. 2020), [Online]. Available: https://pyimagesearch.com/2020/12/21/detecting-aruco-markers-with-opencv-and-python/ (visited on 05/23/2022).

[108] G. Bradski. "Detection of ArUco markers." (2022), [Online]. Available: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html (visited on 05/23/2022).

[109] Sagi Eppel, Haoping Xu, Mor Bismuth, and Alan Aspuru-Guzik, "Computer vision for recognition of materials and vessels in chemistry lab settings and the vector-LabPics data set," *ACS Central Science*, vol. 6, no. 10, pp. 1743–1752, Sep. 2020. DOI: 10.1021/acscentsci.0c00460. [Online]. Available: https://doi.org/10.1021%2Facscentsci.0c00460.

[110] Sagi Eppel, *Code and trained model for Computer Vision for Segmentation and Recognition of Materials and Vessels in Chemistry Lab Settings and the Vector-LabPics Dataset*, Mar. 2020. DOI: 10.5281/zenodo.3697767. [Online]. Available: https://doi.org/10.5281/zenodo.3697767.

[111] chuanqi305, Daniel Rodríguez L., PiyalGeorge, *et al.* "Caffe implementation of Google MobileNet SSD detection network, with pretrained weights on VOC0712 and mAP=0.727." (2021), [Online]. Available: https://github.com/chuanqi305/MobileNet-SSD (visited on 07/13/2022).

[112] Wei Liu, Dragomir Anguelov, Dumitru Erhan, *et al.*, "Ssd: Single shot multibox detector," 2015. DOI: 10.48550/ARXIV.1512.02325. [Online]. Available: https://arxiv.org/abs/1512.02325.

[113] Adrian Rosebrock. "Image thresholding." (2022), [Online]. Available: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html (visited on 07/13/2022).

[114] Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, Eds., *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-24574-4. [Online]. Available: https://doi.org/10.1007%2F978-3-319-24574-4.

[115] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, IEEE, Oct. 2016. DOI: 10.1109/3dv.2016.79. [Online]. Available: https://doi.org/10.1109%2F3dv.2016.79.

[116] John Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986. DOI: 10.1109/tpami.1986.4767851. [Online]. Available: https://doi.org/10.1109%2Ftpami.1986.4767851.

[117] Ikram Morcos, "Determination of surface tensions of liquids from liquid meniscus rise on partially immersed plates," *The Journal of Chemical Physics*, vol. 55, no. 8, pp. 4125–4127, Oct. 1971. DOI: 10.1063/1.1676712. [Online]. Available: https://doi.org/10.1063%2F1.1676712.

[118] Mark White, "A powder and liquid dispensing apparatus," British pat. 2502091A, Nov. 20, 2013. [Online]. Available: https://patents.google.com/patent/GB2502091A/en (visited on 05/31/2022).

[119] John Comley, "Automation of solid/powder dispensing: Much needed, but cautiously used!," vol. 10, pp. 39–51, Jun. 2009.

[120] Standard Educational Corporation, *New Standard Encyclopedia* (New Standard Encyclopedia v. 1). Standard Educational Corporation, 1978, p. A–257, "The Archimedes' screw was developed in ancient Egypt and was subsequently used by Archimedes (287–212 b.c.)", ISBN: 9780873921831. [Online]. Available: https://books.google.ca/books?id=47zWGRWx%5C_y0C (visited on 05/30/2022).

[121] Ultimaker BV. "Ultimaker PP." (2022), [Online]. Available: https://ultimaker.com/materials/pp (visited on 05/30/2022).

[122] McMaster-Carr. "Chemical-resistant PPS rods." (2022), [Online]. Available: https://www.mcmaster.com/machinable-plastics/performance-properties~chemical-resistant/chemical-resistant-pps-rods/ (visited on 05/30/2022).

[123] McMaster-Carr. "Plastic press-fit binding barrels and screws." (2022), [Online]. Available: https://www.mcmaster.com/90249A750/ (visited on 05/23/2022).

[124] Mettler Toledo Group, *WKC205C precision weigh modules*, 2020.

[125] Peter Polidoro and james Pells. "mettler_toledo_device_python." (2021), [Online]. Available: https://github.com/janelia-pypi/mettler_toledo_device_python (visited on 05/25/2022).

[126] Chris Beaven, Mariana Bedran Lesche, cryptogun, *et al.* "Python QR code image generator." (2022), [Online]. Available: https://github.com/lincolnloop/python-qrcode (visited on 05/19/2022).

[127] Kentaro Fukuchi, d4ndo, Sebastian Buchwald, Colin Kennedy, Michał Górny, and Dan Storm. "A fast and compact QR code encoding library." (2020), [Online]. Available: https://github.com/fukuchi/libqrencode/ (visited on 05/19/2022).

[128] Keyence Corporation of America. "Sensors and Machine Vision Systems for Factory Automation." (2022), [Online]. Available: https://www.keyence.com/ (visited on 05/19/2022).

[129] Cognex Corporation. "Cognex | Machine Vision and Barcode Readers." (2022), [Online]. Available: https://www.cognex.com/ (visited on 05/19/2022).

[130] Banner Engineering Corp. "Banner Engineering." (2022), [Online]. Available: https://bannerengineering.com/us/en.html (visited on 05/19/2022).

[131] Raspberry Pi Foundation. "Raspberry Pi Camera Module 2." (2022), [Online]. Available: https://www.raspberrypi.com/products/camera-module-v2/ (visited on 05/19/2022).

[132] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[133] Alexander Alekhin, Vladimir, Pavel Rojtberg, *et al.* "Repository for OpenCV's extra modules." (2022), [Online]. Available: https://github.com/opencv/opencv_contrib (visited on 05/19/2022).

[134] Junhao Liang and Tianqi Wang. "Recognizing one-dimensional barcode using OpenCV." (2021), [Online]. Available: https://opencv.org/recognizing-one-dimensional-barcode-using-opencv/ (visited on 05/19/2022).

[135] Vikas Gupta. "OpenCV QR Code Scanner (C++ and Python)." (2018), [Online]. Available: https://learnopencv.com/opencv-qr-code-scanner-c-and-python/ (visited on 05/19/2022).

[136] G. Bradski. "cv::QRCodeDetector Class Reference." (2022), [Online]. Available: https://docs.opencv.org/4.x/de/dc3/classcv_1_1QRCodeDetector.html (visited on 05/19/2022).

[137] Lawrence Hudson, Ismael Benito, Dmytro Ferens, and Karthikeyan Singaravelan. "Read one-dimensional barcodes and QR codes from python 2 and 3." (2021), [Online]. Available: https://github.com/NaturalHistoryMuseum/pyzbar/ (visited on 05/19/2022).

[138] Jeff Brown. "ZBar bar code reader." (2011), [Online]. Available: http://zbar.sourceforge.net/ (visited on 05/19/2022).

[139] The Pi Hut. "Camera board for raspberry pi - adjustable-focus lens (5mp)." (2022), [Online]. Available: https://thepihut.com/products/raspberry-pi-camera-adjustable-focus-5mp (visited on 07/11/2022).

[140] Arducam. "Pi Hawk-eye: 64MP Ultra High-Res Camera for Raspberry Pi." (2022), [Online]. Available: https://www.arducam.com/64mp-ultra-high-res-camera-raspberry-pi/ (visited on 07/11/2022).

[141] Raspberry Pi Foundation. "Buy a raspberry pi high quality camera." (2022), [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/ (visited on 07/11/2022).

[142] Arducam. "Focal Length Calculator and How Embedded Cameras Work." (2022), [Online]. Available: https://www.arducam.com/focal-length-calculator/ (visited on 07/11/2022).

[143] Brother. "Brother QL-710W | MobilePrintersLabelers | By Brother." (2022), [Online]. Available: https://www.brother-usa.com/products/ql710w (visited on 05/19/2022).

[144] Philipp Klaus, bmanifold, Dan, michieldwitte, Russ Garrett, and jleguen. "Python package for the raster language protocol of the brother ql series label printers (ql-500, ql-550, ql-560, ql-570, ql-700, ql-710w, ql-720nw, ql-800, ql-810w, ql-820nwb, ql-1050, ql-1060n and more)." (2020), [Online]. Available: https://github.com/pklaus/brother_ql (visited on 05/19/2022).

[145] SQLite. "SQLite home page." (2022), [Online]. Available: https://www.sqlite.org/index.html (visited on 07/11/2022).

[146] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec, "The many faces of publish/subscribe," *ACM computing surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

[147] PlatformIO. "PlatformIO Core (CLI) — PlatformIO latest documentation." (2022), [Online]. Available: https://docs.platformio.org/en/latest/core/index.html (visited on 05/05/2022).

[148] JeeLabs. "Use any STM Nucleo as programmer." (2016), [Online]. Available: https://jeelabs.org/book/1547a/index.html (visited on 05/05/2022).

[149] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, Apr. 1950. DOI: 10.1002/j.1538-7305.1950.tb00463.x. [Online]. Available: https://doi.org/10.1002%2Fj.1538-7305.1950.tb00463.x.

[150] Charles J. Fraser, "Electrical and electronics principles," in *Mechanical Engineer's Reference Book*, Elsevier, 1994, pp. 2–1–2–57. DOI: 10.1016/b978-0-7506-1195-4.50006-3. [Online]. Available: https://doi.org/10.1016%2Fb978-0-7506-1195-4.50006-3.

[151] Arm Limited. "Timeout - API references and tutorials | Mbed OS 6 Documentation." (2022), [Online]. Available: https://os.mbed.com/docs/mbed-os/v6.15/apis/timeout.html (visited on 05/05/2022).

[152] Jose Roman Garcia Martinez, Juvenal Rodriguez Resendiz, Miguel Angel Martinez Prado, and Edson Eduardo Cruz Miguel, "Assessment of jerk performance s-curve and trapezoidal velocity profiles," in *2017 XIII International Engineering Congress (CONIIN)*, IEEE, May 2017. DOI: 10.1109/coniin.2017.7968187. [Online]. Available: https://doi.org/10.1109%2Fconiin.2017.7968187.

[153] M Y Stoychitch, "Generate stepper motor linear speed profile in real time," *IOP Conference Series: Materials Science and Engineering*, vol. 294, p. 012 055, Jan. 2018. DOI: 10.1088/1757-899x/294/1/012055. [Online]. Available: https://doi.org/10.1088%2F1757-899x%2F294%2F1%2F012055.

[154] Pramod Ranade. "Linear motor control without the math." (Jan. 2009), [Online]. Available: https://www.eetimes.com/linear-motor-control-without-the-math/ (visited on 05/10/2022).

[155] Embedded staff. "Generate stepper-motor speed profiles in real time." (Dec. 2004), [Online]. Available: https://www.embedded.com/generate-stepper-motor-speed-profiles-in-real-time/ (visited on 05/10/2022).

[156] P.J. Siripala and Y. Ahmet Sekercioglu, "A generalised solution for generating stepper motor speed profiles in real time," *Mechatronics*, vol. 23, no. 5, pp. 541–547, Aug. 2013. DOI: 10.1016/j.mechatronics.2013.04.004. [Online]. Available: https://doi.org/10.1016%2Fj.mechatronics.2013.04.004.

[157] Sichun Xu. "Constant angular acceleration of a stepper motor with arduino." (Apr. 2021), [Online]. Available: https://www.youtube.com/watch?v=Im2Dnc9_Ifo (visited on 05/10/2022).

[158] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[159] Mozilla Developer Network Contributors. "The websocket api (websockets)." (2022), [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (visited on 05/11/2022).

[160] Opentrons. "Opentrons Python API V2 Documentation." (2022), [Online]. Available: https://docs.opentrons.com/v2/index.html (visited on 05/19/2022).

[161] Mike Cousins. "Feat: Safe-to-move signal for custom modules · Comment on GitHub Issue #7619 · Opentrons/opentrons." (2021), [Online]. Available: https://github.com/Opentrons/opentrons/issues/7619#issuecomment-818026103 (visited on 05/10/2022).

[162] Python Software Foundation. "3. data model — python 3.10.4 documentation." (May 2022), [Online]. Available: https://docs.python.org/3/reference/datamodel.html (visited on 05/10/2022).

[163] Meta Platforms, Inc. "React - a javascript library for building user interfaces." (2022), [Online]. Available: https://reactjs.org/ (visited on 05/19/2022).

[164] OpenJS Foundation. "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." (2022), [Online]. Available: https://nodejs.org/en/ (visited on 05/19/2022).

[165] CSA Group. "CAN/CSA-Z434-14 industrial robots and robot systems." (2019), [Online]. Available: https://www.csagroup.org/store/product/CAN%25100CSA-Z434-14/ (visited on 05/22/2022).

[166] Universal Robots. "The UR5e - a flexible collaborative robot arm." (2022), [Online]. Available: https://www.universal-robots.com/products/ur5-robot/ (visited on 05/23/2022).

[167] FANUC America Corporation. "CR-4iA Collaborative Robot." (2022), [Online]. Available: https://www.fanucamerica.com/products/robots/series/collaborative-robot/cr-4ia-cobot (visited on 05/23/2022).

[168] KUKA AG. "LBR iiwa." (2022), [Online]. Available: https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa (visited on 05/23/2022).

[169] Waterloo RoboHub, University of Waterloo. "Welcome to the home of robotics at Waterloo." (2021), [Online]. Available: https://uwaterloo.ca/robohub/ (visited on 05/23/2022).

[170] Franka Emika. "Next Generation Robotics." (2022), [Online]. Available: https://www.franka.de/ (visited on 05/23/2022).

[171] POMO Robotics. "Franka Emika Panda." (2022), [Online]. Available: https://www.pomorobotics.com/robots/frankpanda/ (visited on 05/23/2022).

[172] WiredWorkers. "Franka Emika Panda Collaborative Lightweight Robot." (2022), [Online]. Available: https://wiredworkers.io/cobot-brands/cobot-franka-emika-panda/ (visited on 05/23/2022).

[173] Waterloo RoboHub, University of Waterloo. "Panda Powertool." (2022), [Online]. Available: https://uwaterloo.ca/robohub/people-profiles/panda-powertool (visited on 05/23/2022).

[174] Franka Emika GmbH. "The Robot System." (2021), [Online]. Available: https://www.franka.de/robot-system/ (visited on 11/14/2021).

[175] Clearpath Robotics Inc. "Mobile Robots for Research & Development." (2022), [Online]. Available: https://clearpathrobotics.com/ (visited on 05/23/2022).

[176] Franka Emika. "Franka Control Interface Documentation — Franka Control Interface (FCI) documentation." (2022), [Online]. Available: https://frankaemika.github.io/docs/ (visited on 07/15/2022).

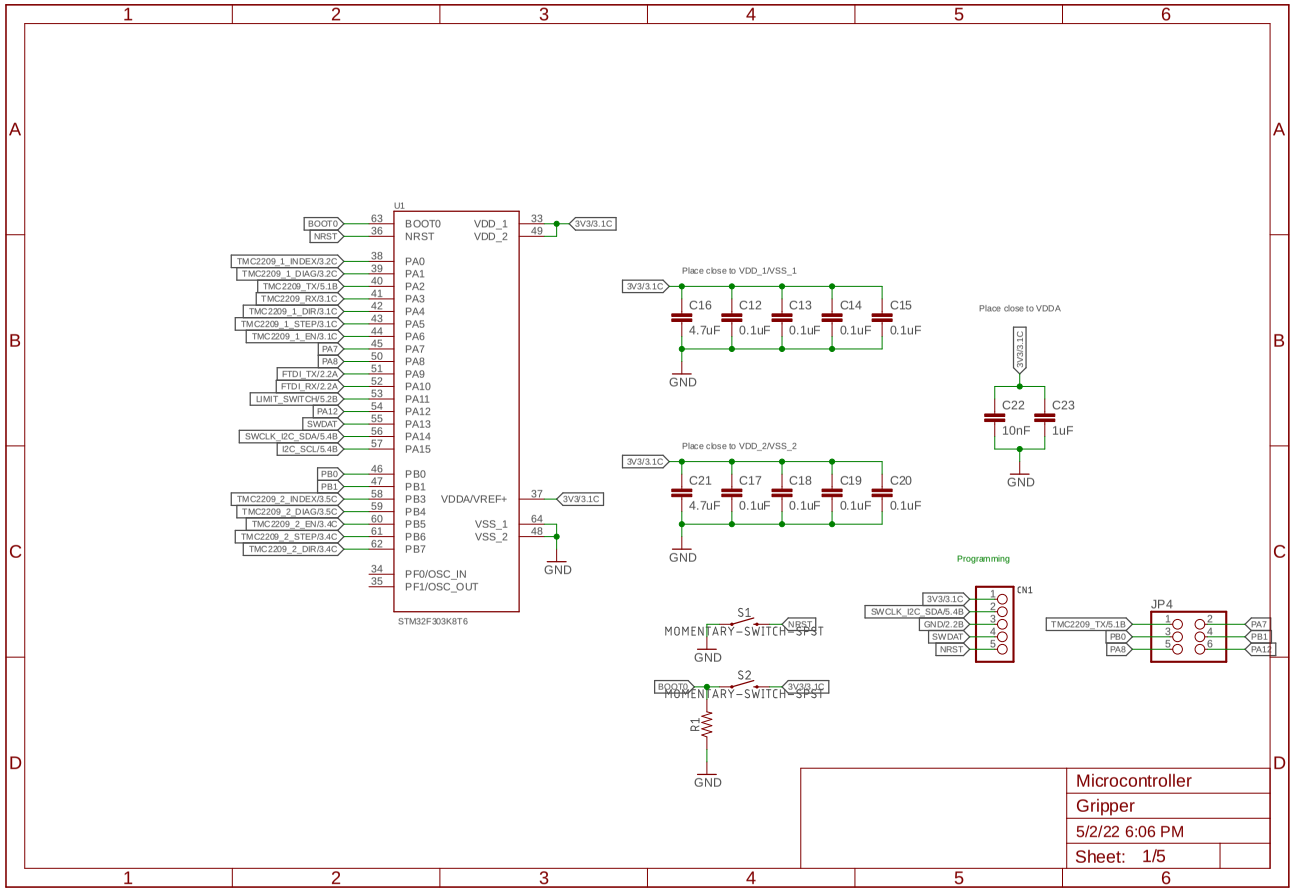
[177] Franka Emika. "Ride Documentation — ride-cli documentation." (2022), [Online]. Available: https://download.franka.de/ride-cli-doc/index.html (visited on 07/15/2022).

[178] Saunab Ghosh, Sergei M. Bachilo, and R. Bruce Weisman, "Supplementary information for advanced sorting of single-walled carbon nanotubes by nonlinear density-gradient ultracentrifugation," *Nature Nanotechnology*, vol. 5, no. 6, pp. 443–450, May 2010. DOI: 10.1038/nnano.2010.68. [Online]. Available: https://doi.org/10.1038%2Fnnano.2010.68.

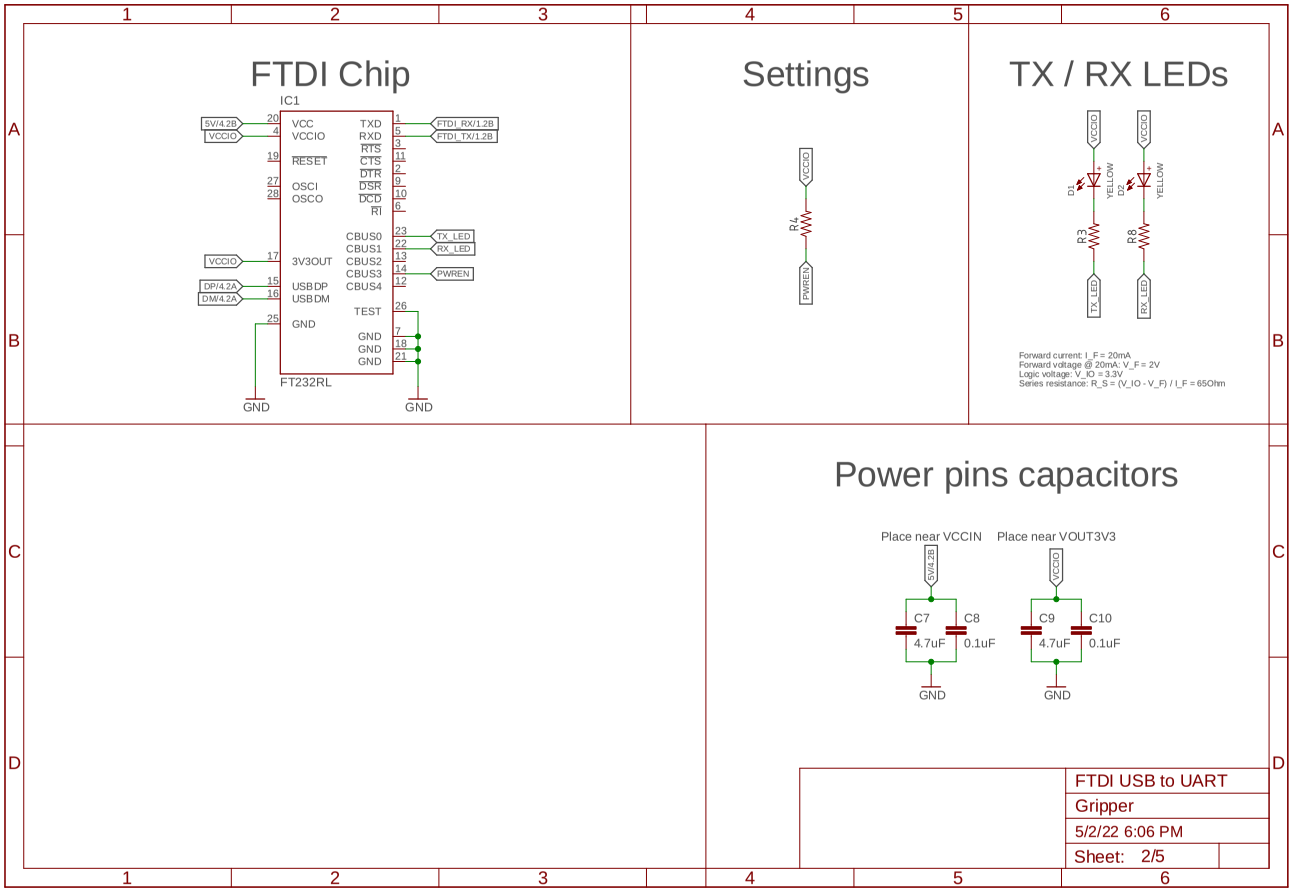
Appendices

Appendix A

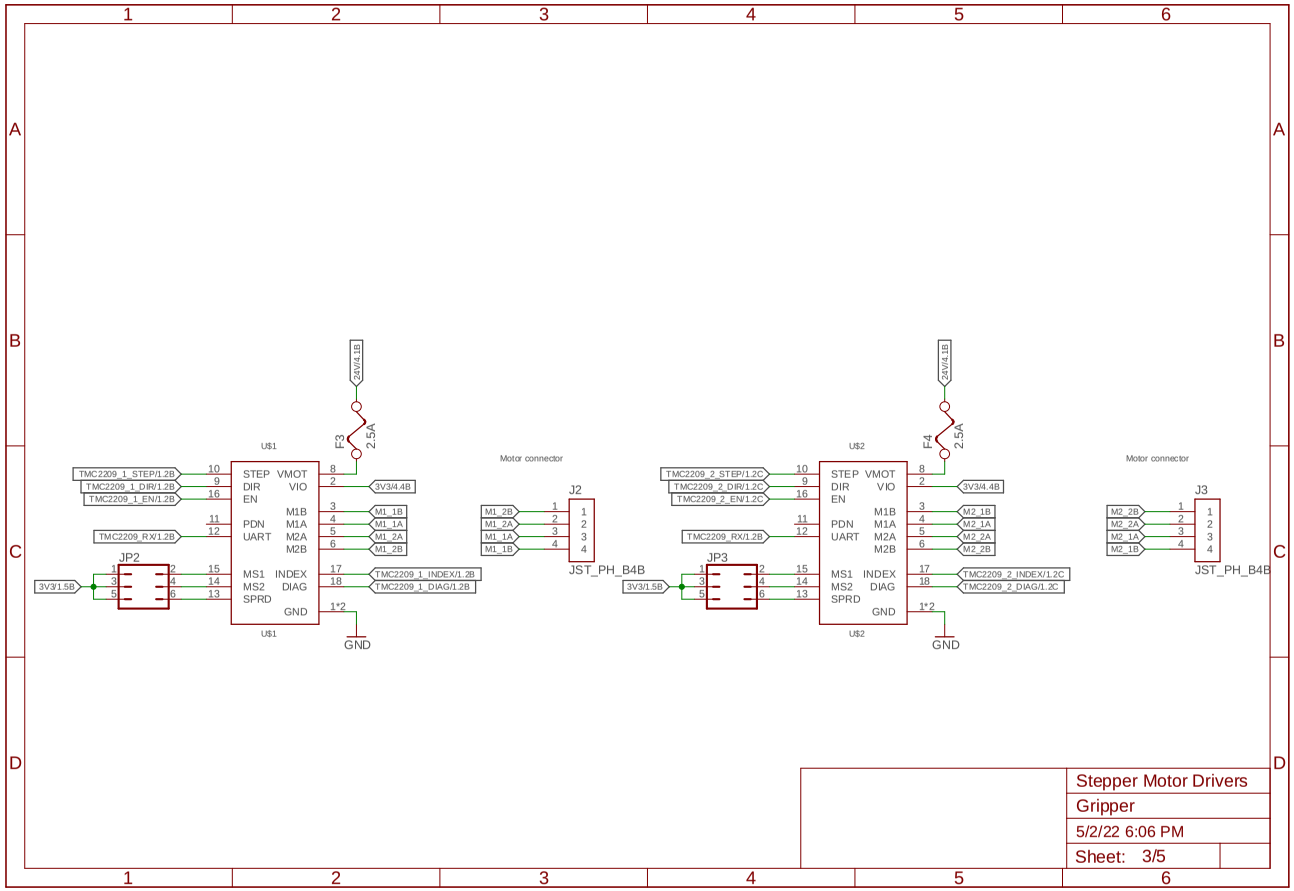
Schematic of the *Gripper V2* PCB in *Autodesk Eagle*

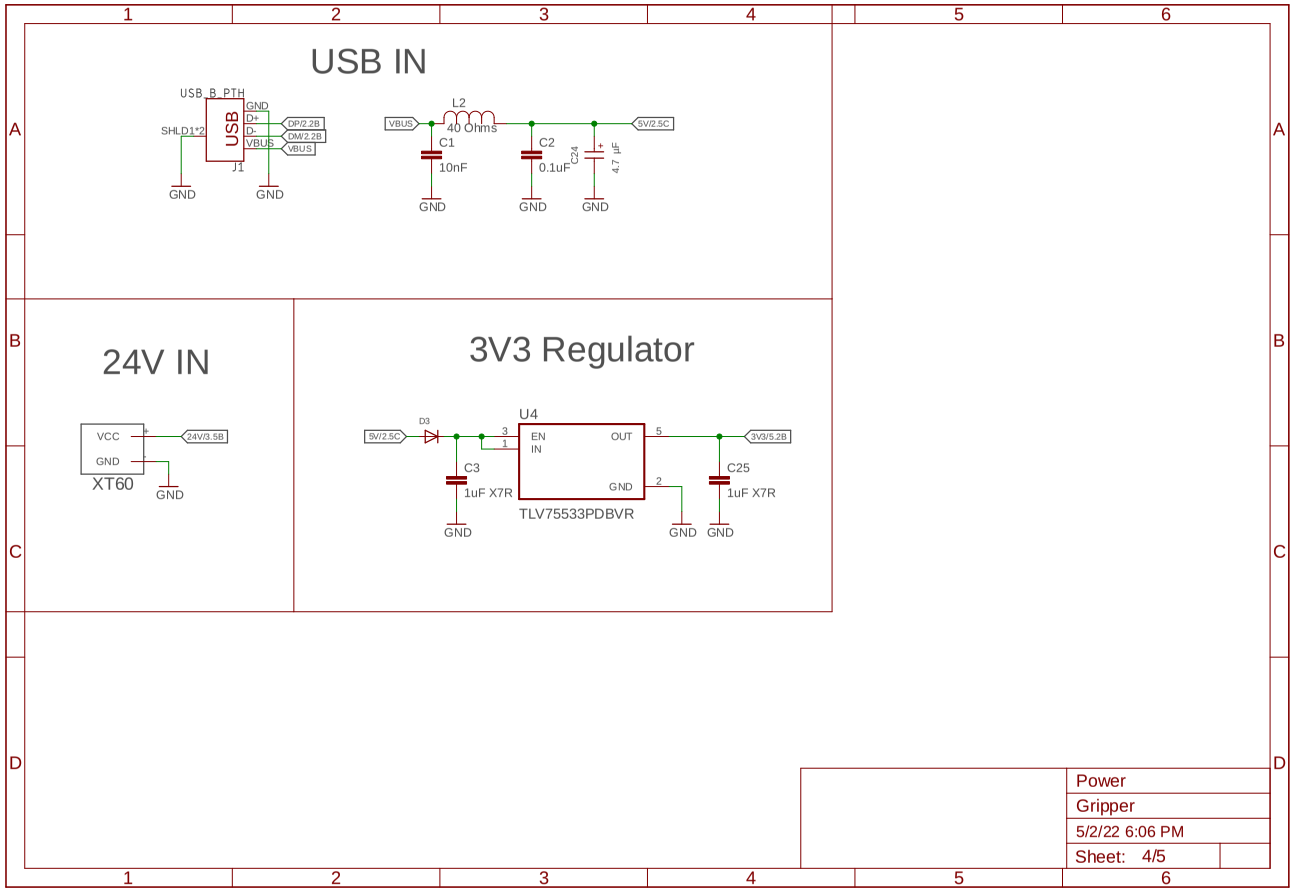


5/2/22 6:13 PM f=0.99 /home/marcel/EAGLE/projects/QuIN/CNT Automation/Gripper/Gripper.sch (Sheet: 1/5)

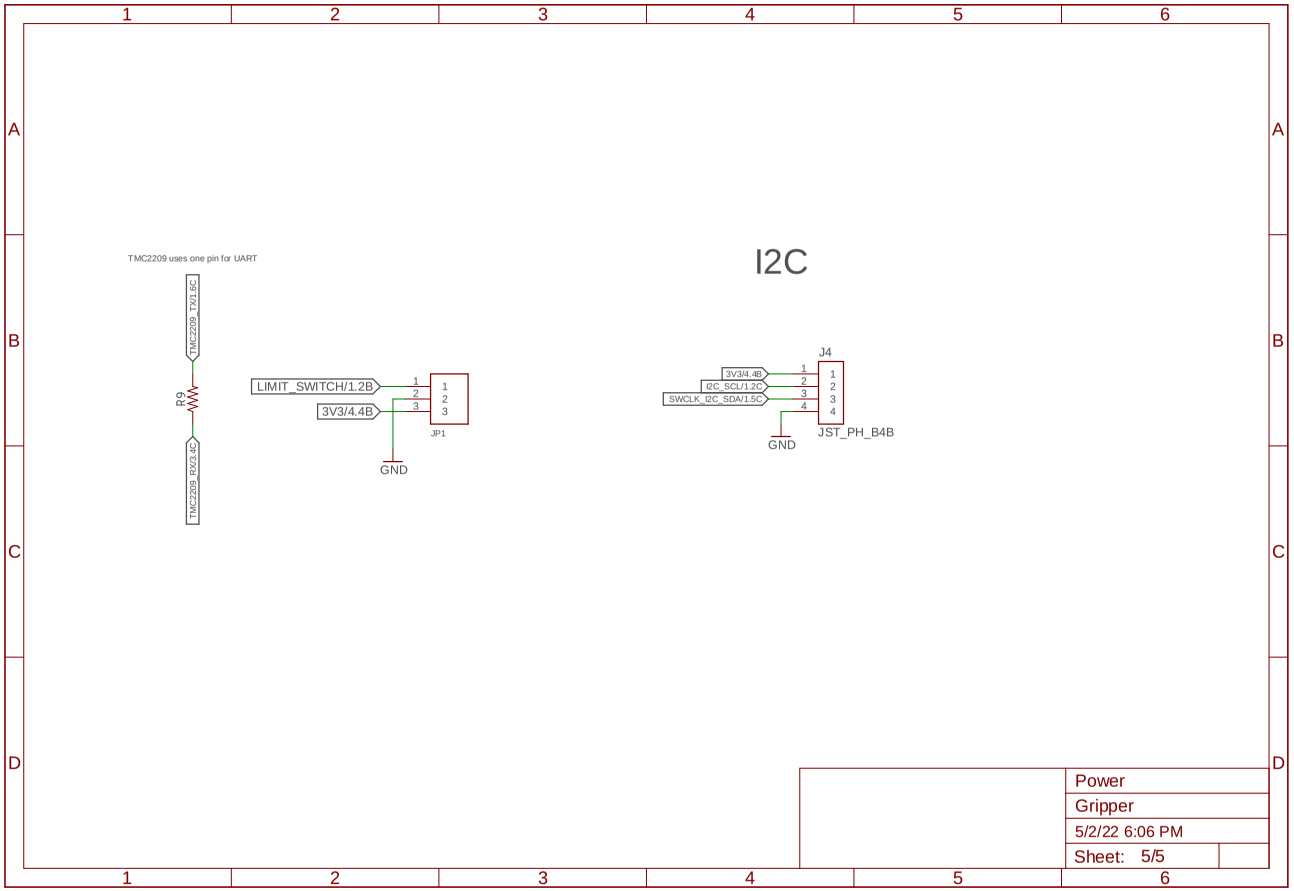


5/2/22 6:13 PM f=0.99 /home/marcel/EAGLE/projects/QuIN/CNT Automation/Gripper/Gripper.sch (Sheet: 2/5)


1
2
3
4
5
6
A
B
C
D
US1
F3
2.5A
24V/4.1B
TMC2209_1_STEP/1.2B
TMC2209_1_DIR/1.2B
TMC2209_1_EN/1.2B
TMC2209_RX/1.2B
STEP
DIR
EN
PDN
UART
VMOT
VIO
M1B
M1A
M2A
M2B
MS1
MS2
SPRD
INDEX
DIAG
GND
3V3/4.4B
M1_1B
M1_1A
M1_2A
M1_2B
TMC2209_1_INDEX/1.2B
TMC2209_1_DIAG/1.2B
1*2
JP2
3V3/1.5B
Motor connector
J2
M1_2B
M1_2A
M1_1A
M1_1B
JST_PH_B4B
US2
F4
TMC2209_2_STEP/1.2C
TMC2209_2_DIR/1.2C
TMC2209_2_EN/1.2C
TMC2209_RX/1.2B
M2_1B
M2_1A
M2_2A
M2_2B
TMC2209_2_INDEX/1.2C
TMC2209_2_DIAG/1.2C
JP3
J3
Stepper Motor Drivers
Gripper
5/2/22 6:06 PM
Sheet: 3/5

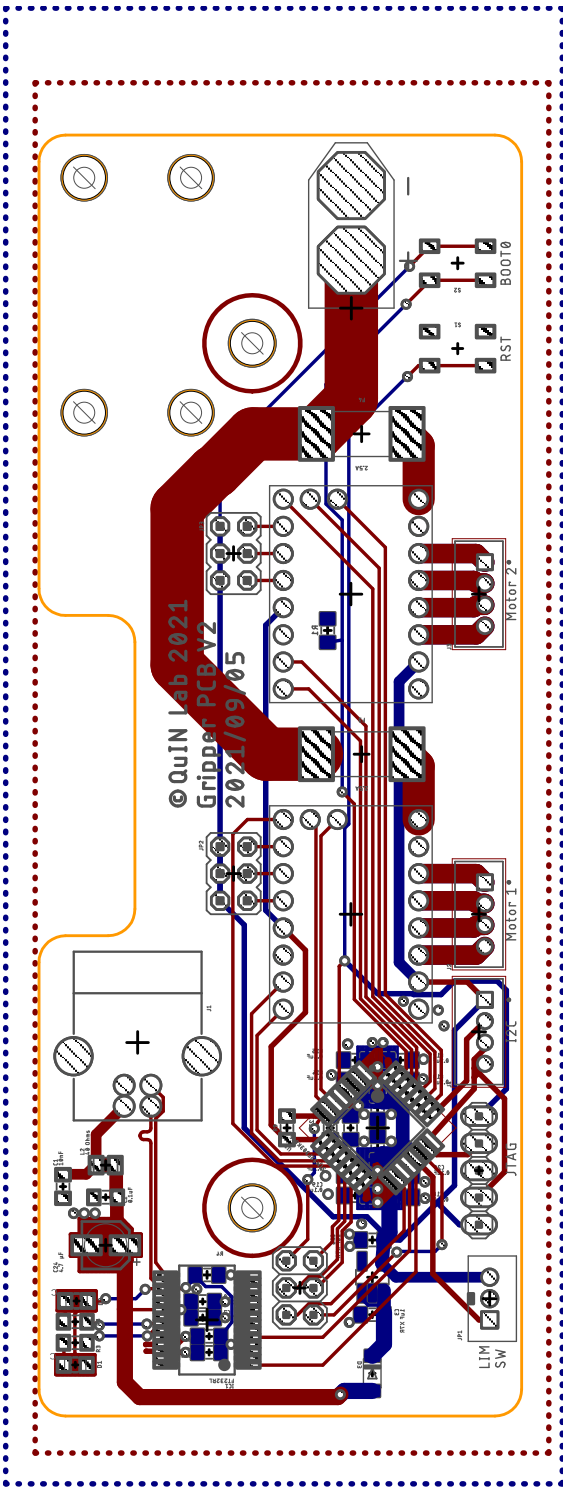


5/2/22 6:13 PM f=0.99 /home/marcel/EAGLE/projects/QuIN/CNT Automation/Gripper/Gripper.sch (Sheet: 4/5)


1
2
3
4
5
6
A
B
C
D
TMC2209 uses one pin for UART
TMC2209 - TXD1.6C
R9
TMC2209 - RXD3.4C
LIMIT_SWITCH/1.2B
3V3/4.4B
GND
JP1
I2C
J4
3V3/4.4B
I2C_SCL/1.2C
SWCLK_I2C_SDA/1.5C
GND
JST_PH_B4B
Power
Gripper
5/2/22 6:06 PM
Sheet: 5/5

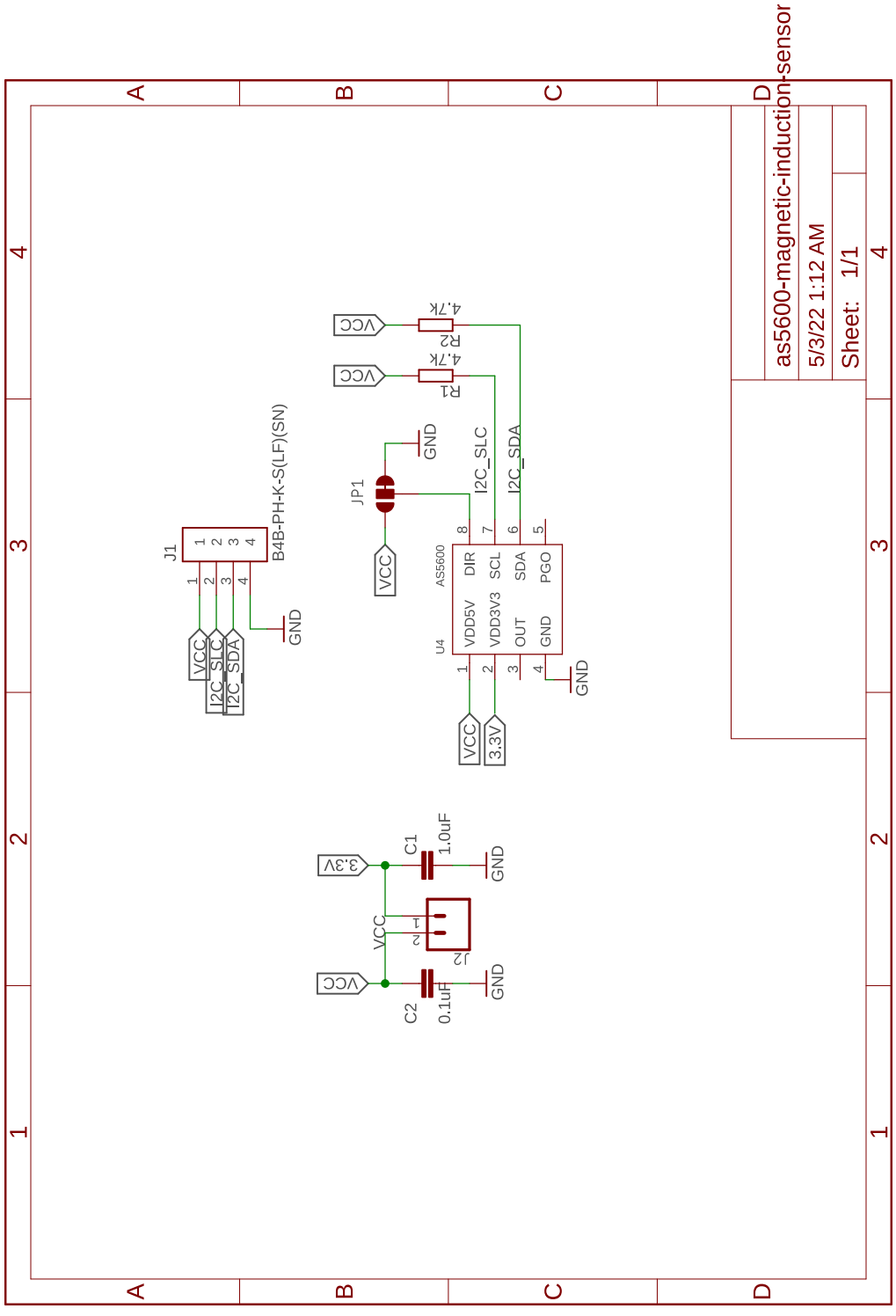
Appendix B

Board of the *Gripper V2* PCB in *Autodesk Eagle*


©QuIN Lab 2021
Gripper PCB V2
2021/09/05
BOOT0
RST
Motor 2*
Motor 1*
I2C
JTAG
LIM SW

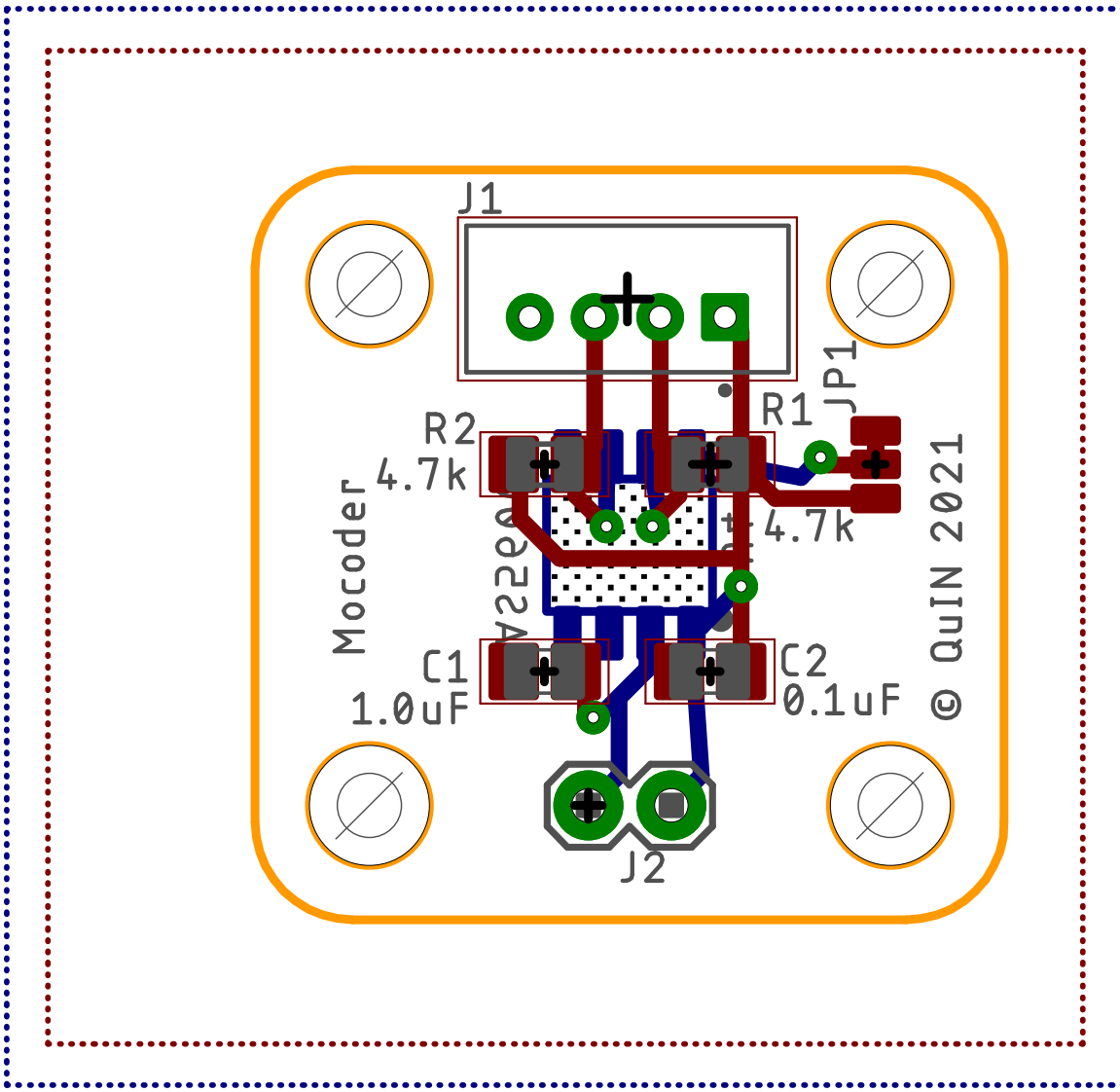
Appendix C

Schematic of the AS5600-based magnetic induction encoder PCB in *Autodesk Eagle*


as5600-magnetic-induction-sensor
5/3/22 1:12 AM
Sheet: 1/1
A
B
C
D
1
2
3
4
J1
B4B-PH-K-S(LF)(SN)
VCC
I2C_SLC
I2C_SDA
GND
JP1
R1
4.7K
R2
4.7K
U4
AS5600
DIR
SCL
SDA
PGO
VDD5V
VDD3V3
OUT
GND
3.3V
C1
1.0uF
C2
0.1uF
J2

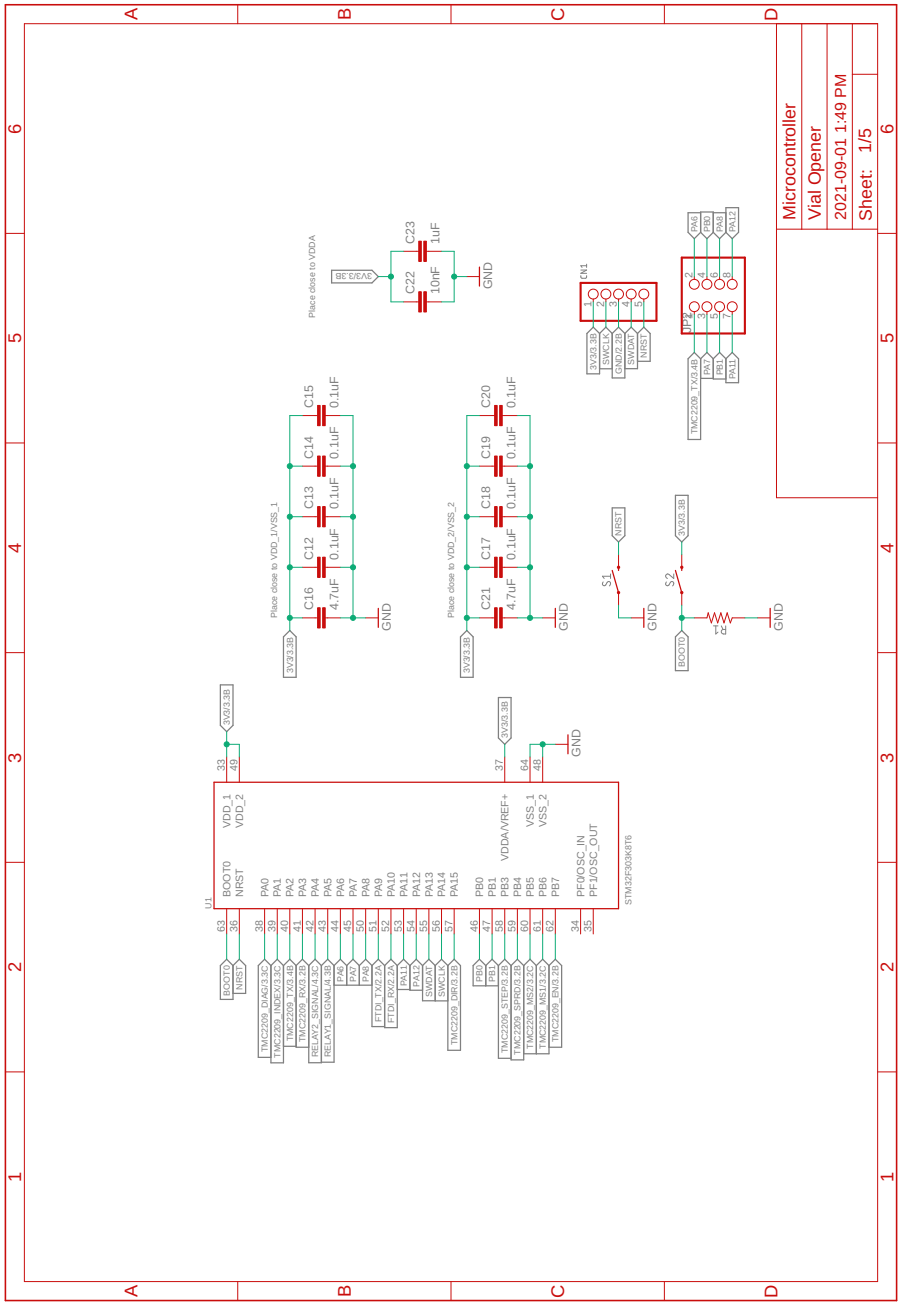
Appendix D

Board of the AS5600-based magnetic induction encoder PCB in *Autodesk Eagle*

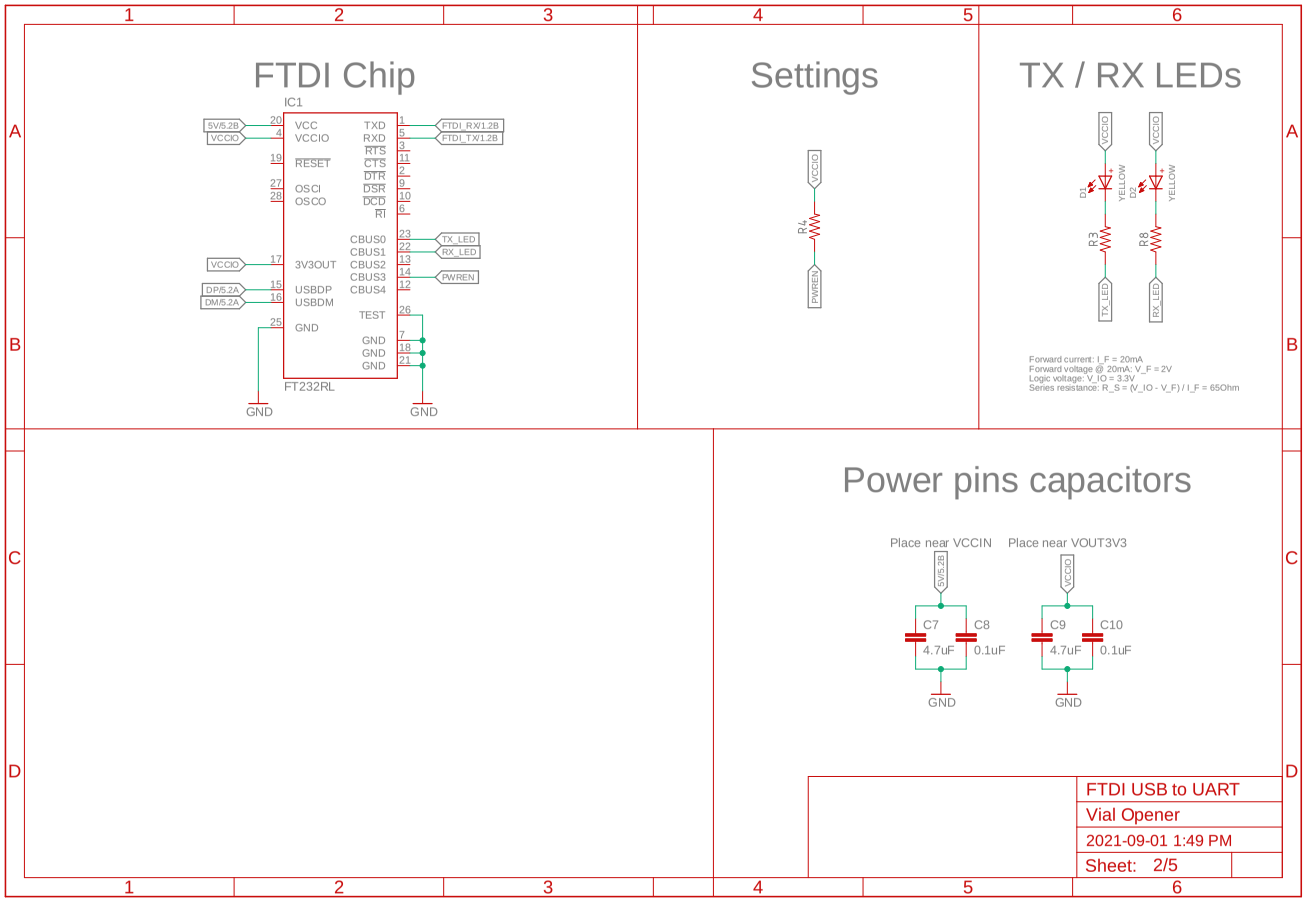

J1
JP1
R1
R2
4.7k
4.7k
Mocoder
A2224
C1
1.0uF
C2
0.1uF
© QuIN 2021
J2

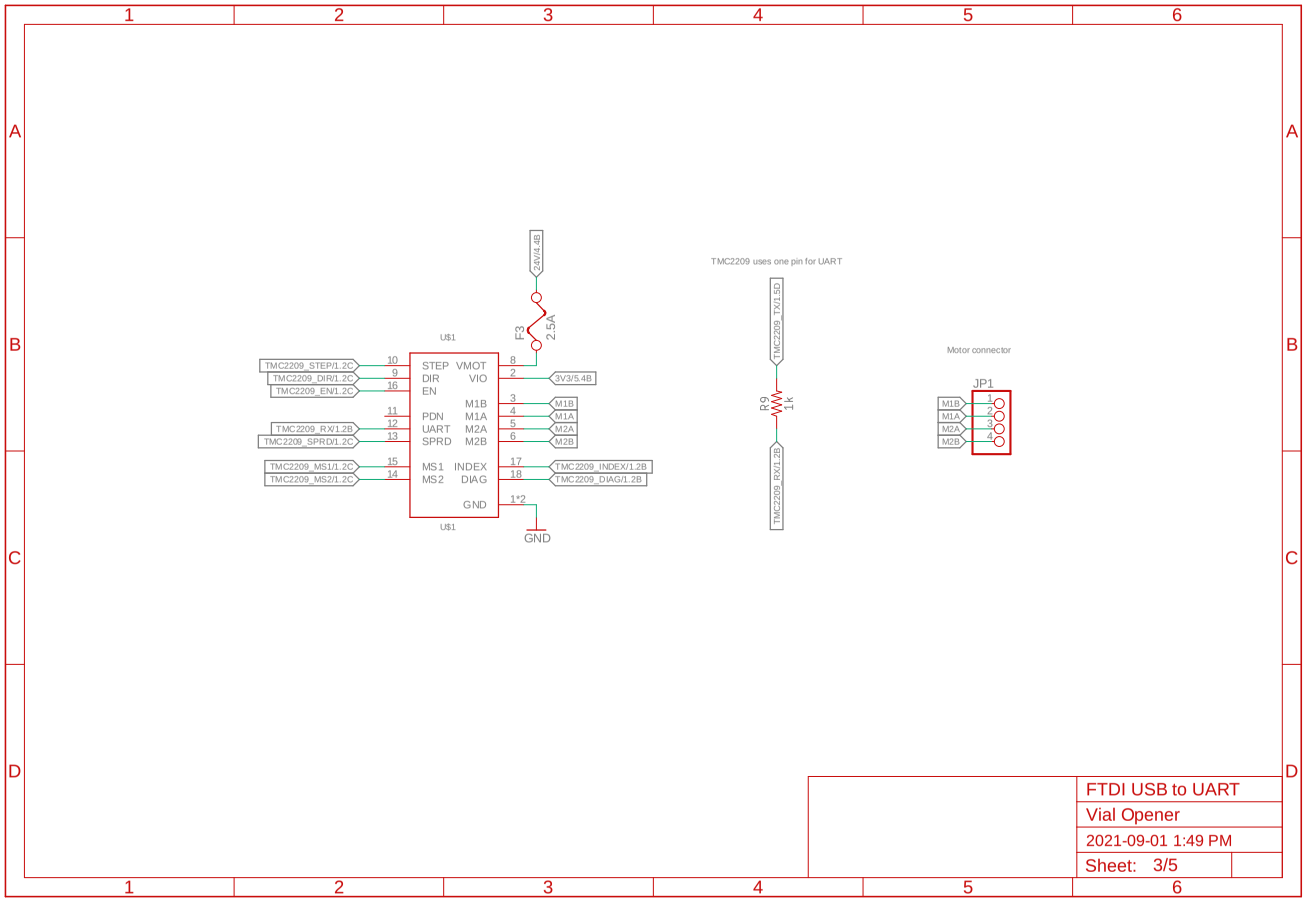
Appendix E

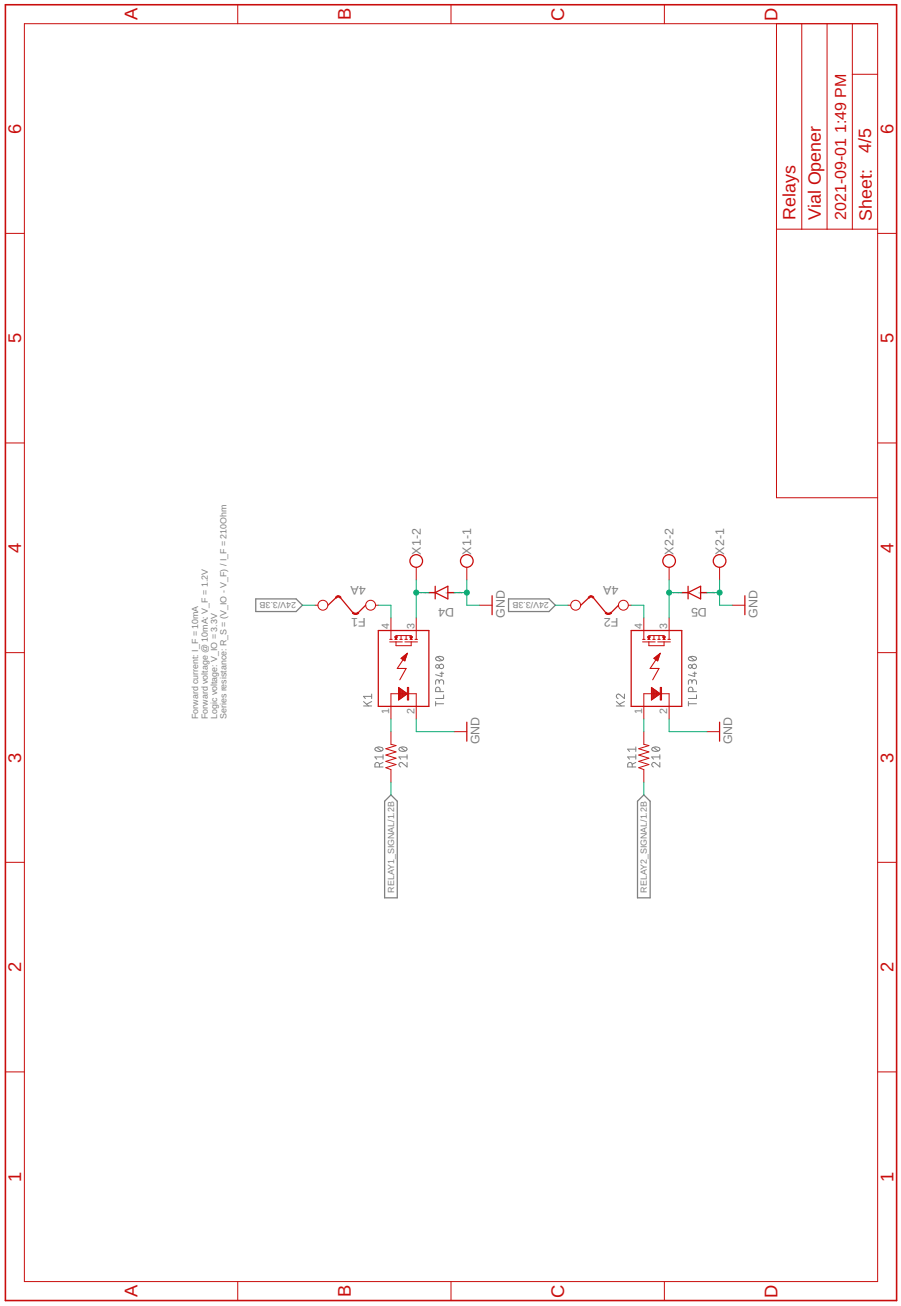
Schematic of the *Vial Opener* PCB in *Autodesk Eagle*



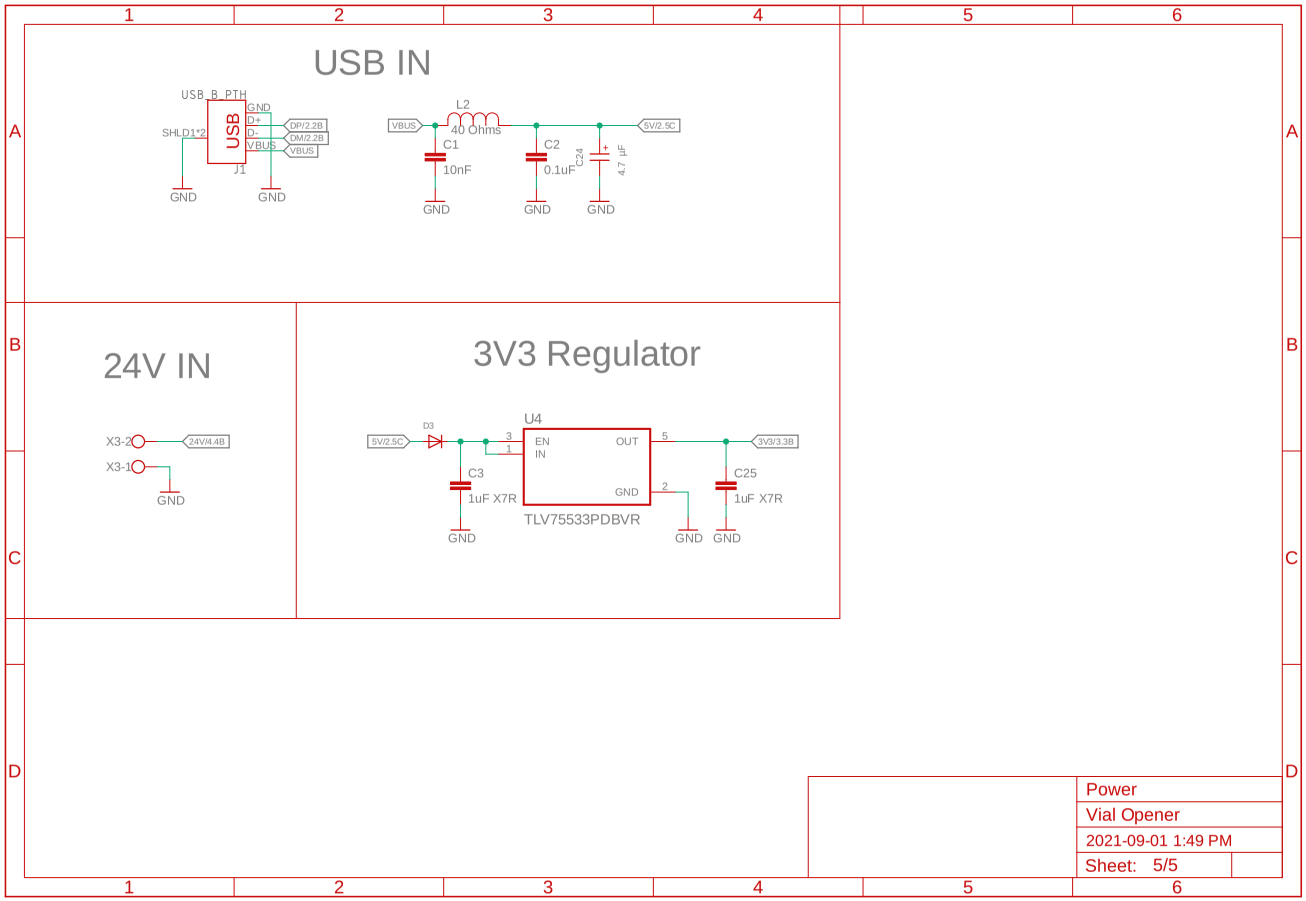
2022-05-22 3:37 AM /home/marcel/EAGLE/projects/QuIN Lab/vial-oener-pcb/Vial Opener.sch (Sheet: 1/5)


FTDI Chip
IC1
5V/5.2B
VCCIO
VCC
VCCIO
RESET
OSCI
OSCO
3V3OUT
USBDP
USBDM
GND
TXD
RXD
RTS
CTS
DTR
DSR
DCD
RI
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
TEST
GND
GND
GND
FTDI_RX/1.2B
FTDI_TX/1.2B
TX_LED
RX_LED
PWREN
DP/5.2A
DM/5.2A
FT232RL
GND
GND
Settings
VCCIO
R4
PWREN
TX / RX LEDs
VCCIO
VCCIO
D1
YELLOW
D2
YELLOW
R3
R8
TX_LED
RX_LED
Forward current: I_F = 20mA
Forward voltage @ 20mA: V_F = 2V
Logic voltage: V_IO = 3.3V
Series resistance: R_S = (V_IO - V_F) / I_F = 65Ohm
Power pins capacitors
Place near VCCIN
Place near VOUT3V3
5V/5.2B
VCCIO
C7
C8
4.7uF
0.1uF
C9
C10
4.7uF
0.1uF
GND
GND
FTDI USB to UART
Vial Opener
2021-09-01 1:49 PM
Sheet: 2/5


1
2
3
4
5
6
A
B
C
D
U$1
STEP
DIR
EN
PDN
UART
SPRD
MS1
MS2
VMOT
VIO
M1B
M1A
M2A
M2B
INDEX
DIAG
GND
TMC2209_STEP/1.2C
TMC2209_DIR/1.2C
TMC2209_EN/1.2C
TMC2209_RX/1.2B
TMC2209_SPRD/1.2C
TMC2209_MS1/1.2C
TMC2209_MS2/1.2C
TMC2209_INDEX/1.2B
TMC2209_DIAG/1.2B
3V3/5.4B
F3
2.5A
TMC2209 uses one pin for UART
R9
1k
TMC2209_TX/1.5D
TMC2209_RX/1.2B
Motor connector
JP1
FTDI USB to UART
Vial Opener
2021-09-01 1:49 PM
Sheet: 3/5

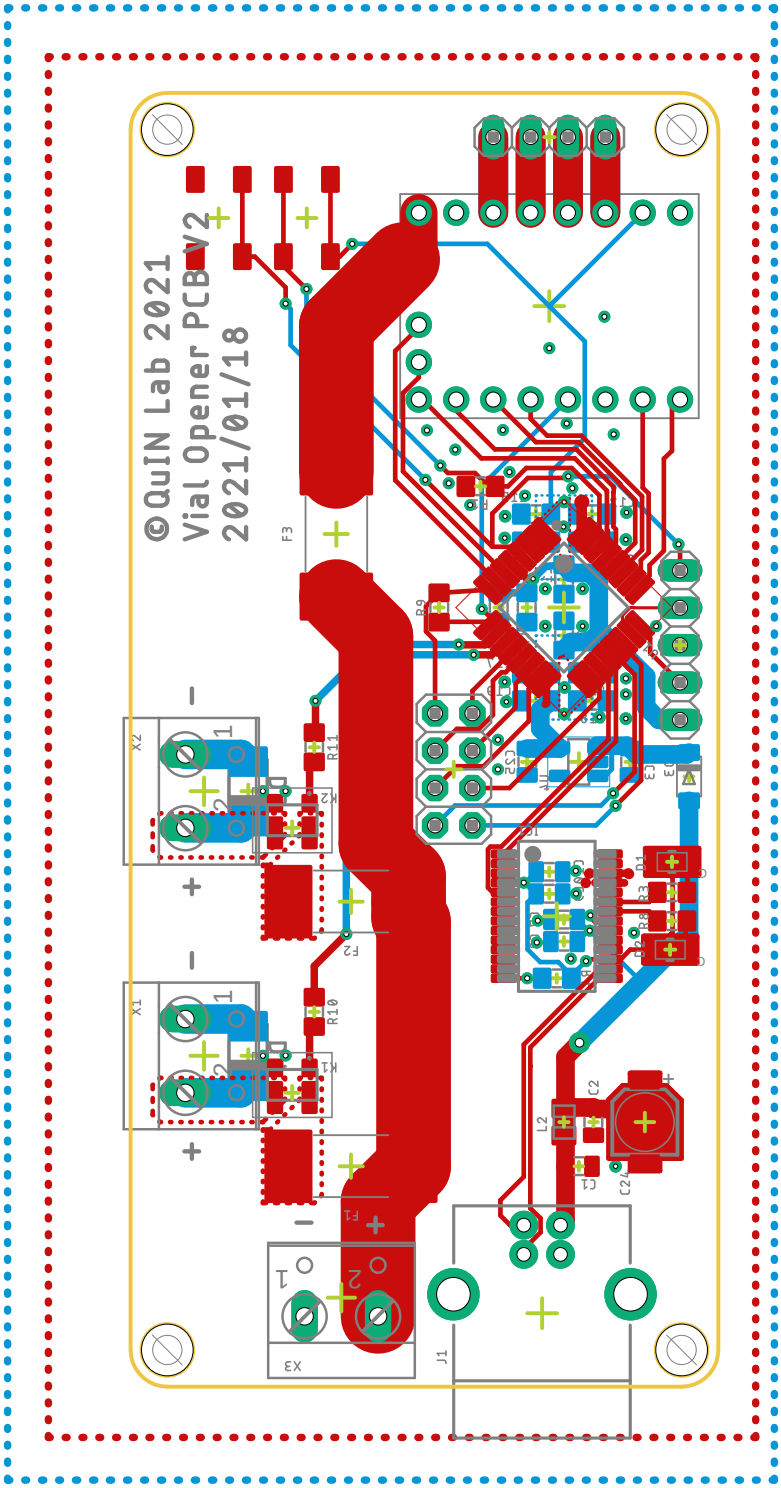


2022-05-22 3:37 AM /home/marcel/EAGLE/projects/QuIN Lab/vial-oener-pcb/Vial Opener.sch (Sheet: 4/5)


USB IN
USB_B_PTH
USB
GND
D+
D-
VBUS
SHLD1*2
J1
DP/2.2B
DM/2.2B
VBUS
GND
GND
L2
40 Ohms
VBUS
5V/2.5C
C1
10nF
C2
0.1uF
C24
4.7 µF
GND
GND
GND
24V IN
X3-2
X3-1
24V/4.4B
GND
3V3 Regulator
U4
D3
5V/2.5C
EN
IN
OUT
GND
3
1
5
2
3V3/3.3B
C3
1uF X7R
C25
1uF X7R
TLV75533PDBVR
GND
GND
GND
1
2
3
4
5
6
A
B
C
D
Power
Vial Opener
2021-09-01 1:49 PM
Sheet: 5/5

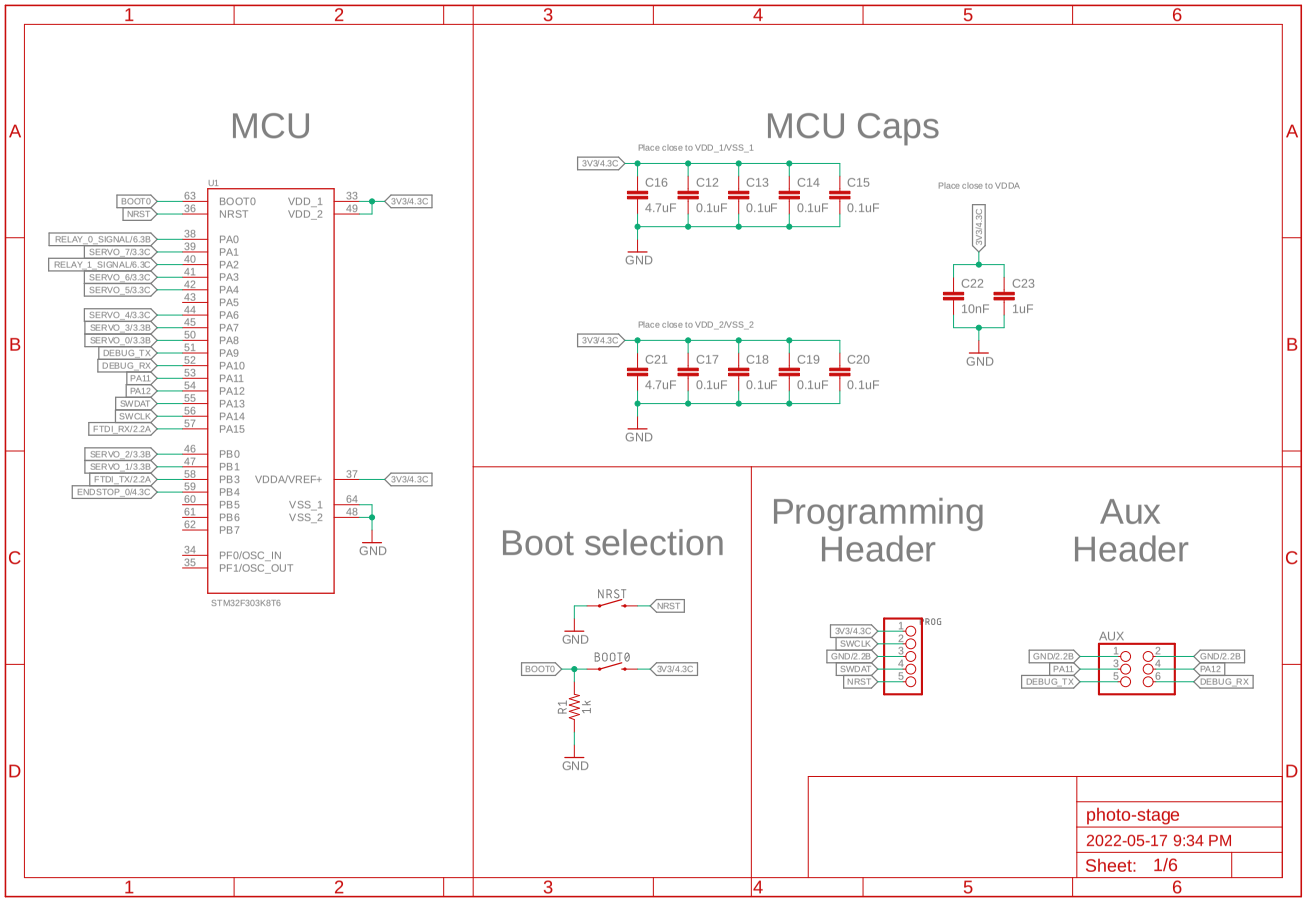
Appendix F

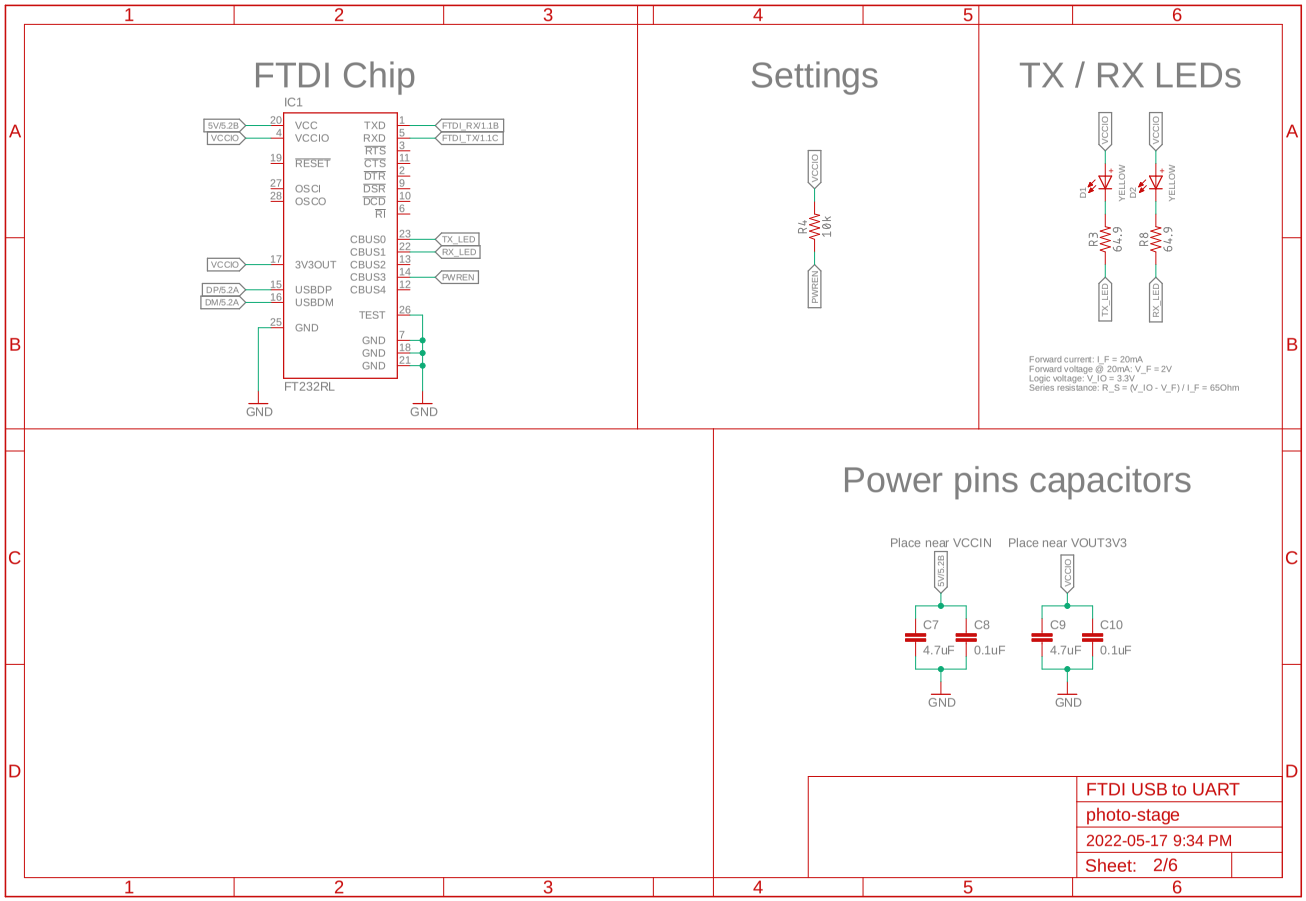
Board of the *Vial Opener* PCB in *Autodesk Eagle*

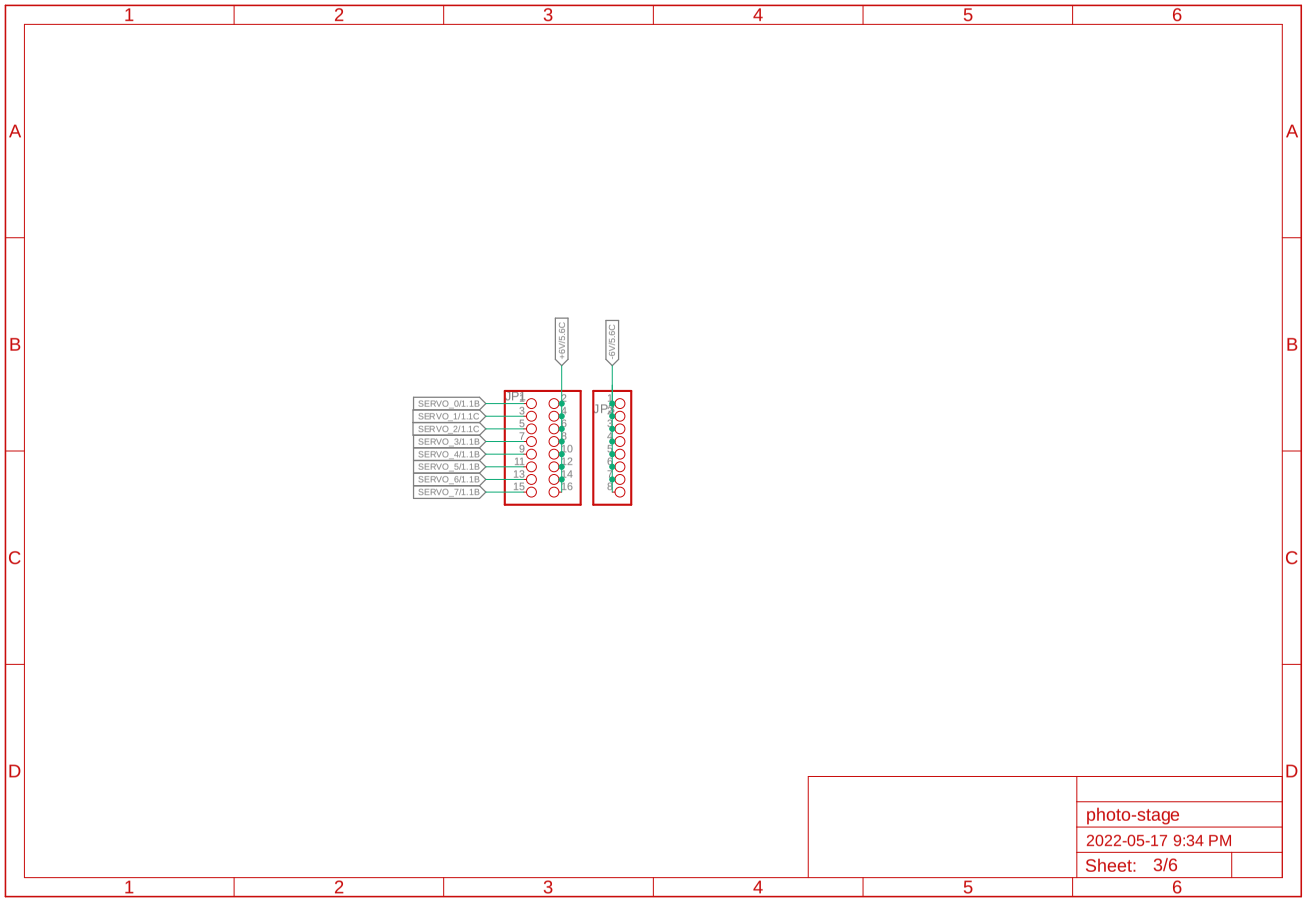

©QuIN Lab 2021
Vial Opener PCB V2
2021/01/18

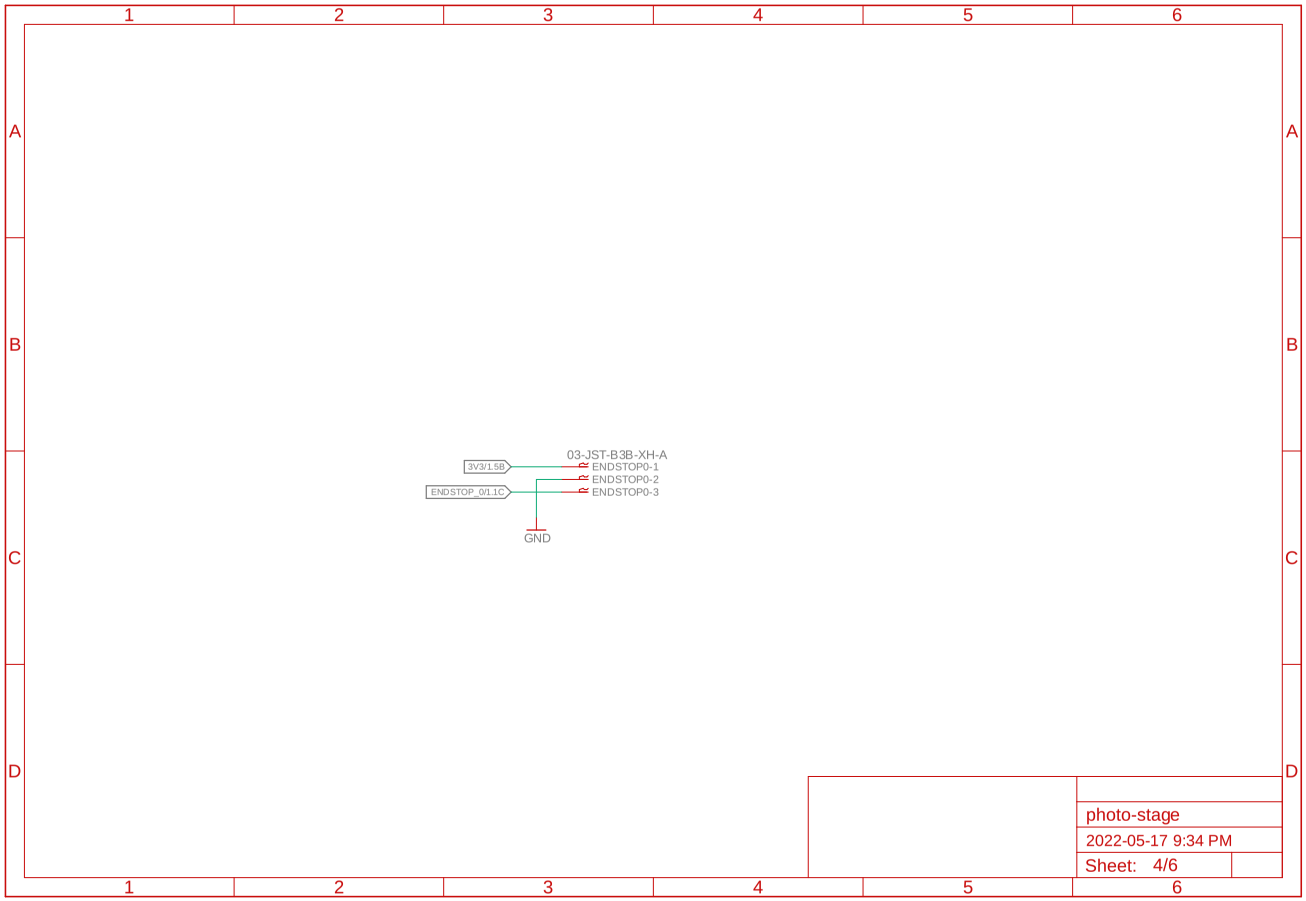
Appendix G

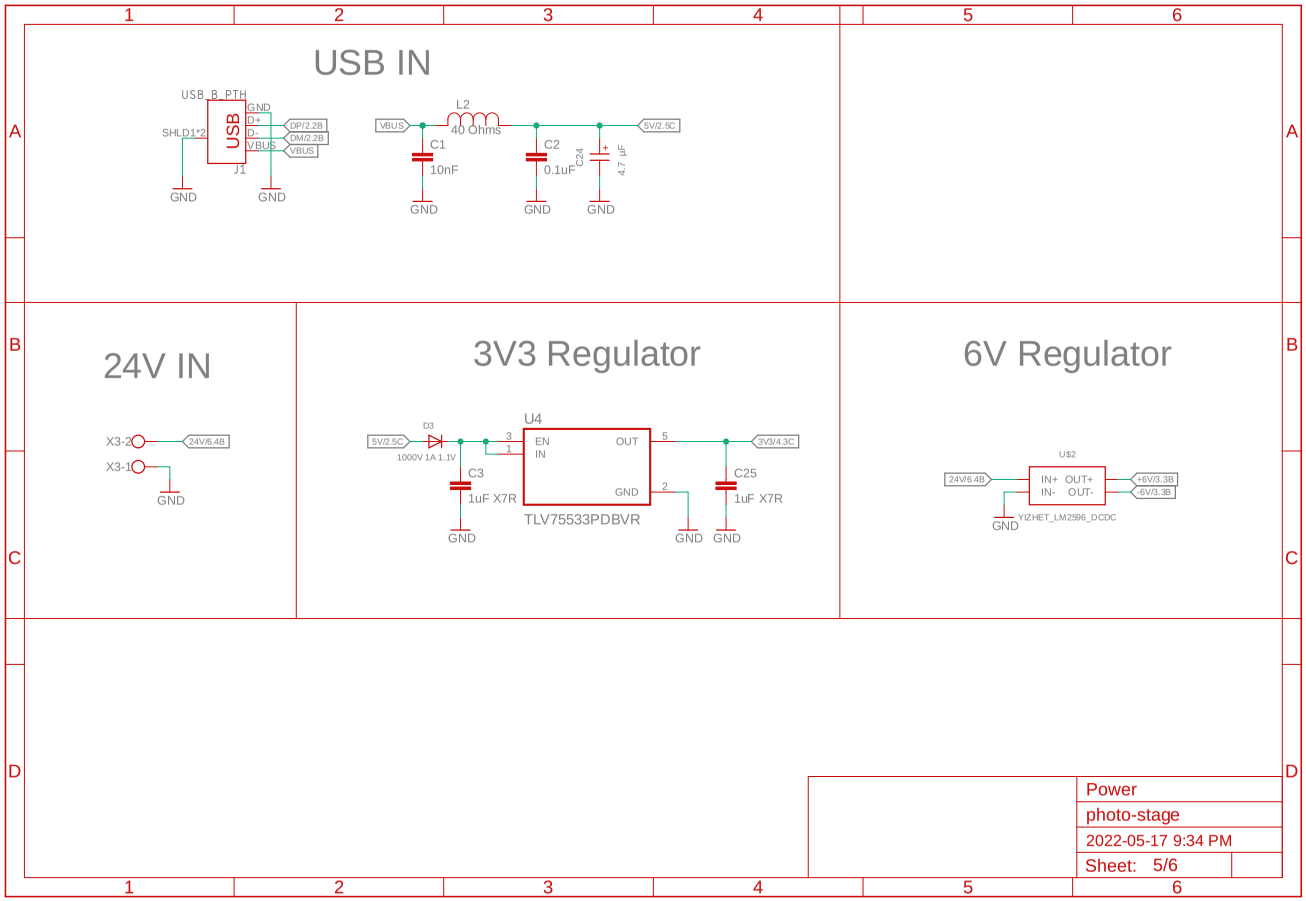
Schematic of the *Photo Stage* PCB in *Autodesk Eagle*

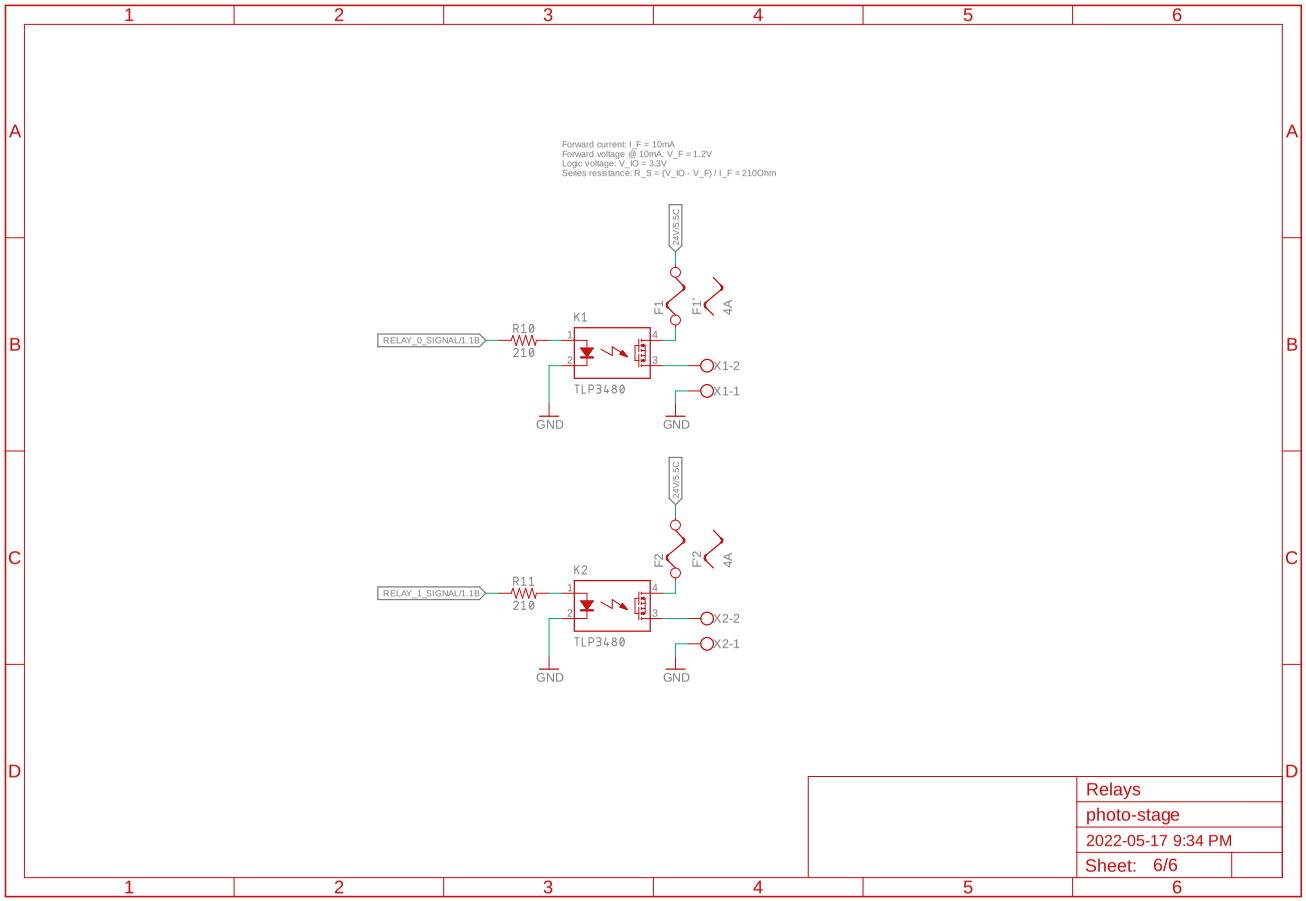

MCU
U1
BOOT0
NRST
VDD_1
VDD_2
3V3/4.3C
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB3
PB4
PB5
PB6
PB7
VDDA/VREF+
VSS_1
VSS_2
GND
PF0/OSC_IN
PF1/OSC_OUT
STM32F303K8T6
RELAY_0_SIGNAL/6.3B
SERVO_7/3.3C
RELAY_1_SIGNAL/6.3C
SERVO_6/3.3C
SERVO_5/3.3C
SERVO_4/3.3C
SERVO_3/3.3B
SERVO_0/3.3B
DEBUG_TX
DEBUG_RX
PA11
PA12
SWDAT
SWCLK
FTDI_RX/2.2A
SERVO_2/3.3B
SERVO_1/3.3B
FTDI_TX/2.2A
ENDSTOP_0/4.3C
MCU Caps
Place close to VDD_1/VSS_1
C16
C12
C13
C14
C15
4.7uF
0.1uF
Place close to VDDA
C22
C23
10nF
1uF
Place close to VDD_2/VSS_2
C21
C17
C18
C19
C20
Boot selection
NRST
BOOT0
R1
1k
Programming Header
PROG
Aux Header
AUX
GND/2.2B
PA11
DEBUG_TX
PA12
DEBUG_RX
photo-stage
2022-05-17 9:34 PM
Sheet: 1/6


FTDI Chip
IC1
5V/5.2B
VCCIO
VCC
VCCIO
RESET
OSCI
OSCO
3V3OUT
USBDP
USBDM
GND
TXD
RXD
RTS
CTS
DTR
DSR
DCD
RI
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
TEST
GND
GND
GND
FTDI_RX/1.1B
FTDI_TX/1.1C
TX_LED
RX_LED
PWREN
DP/5.2A
DM/5.2A
FT232RL
GND
GND
Settings
VCCIO
R4
10K
PWREN
TX / RX LEDs
VCCIO
VCCIO
D1
YELLOW
D2
YELLOW
R3
64.9
R8
64.9
TX_LED
RX_LED
Forward current: I_F = 20mA
Forward voltage @ 20mA: V_F = 2V
Logic voltage: V_IO = 3.3V
Series resistance: R_S = (V_IO - V_F) / I_F = 65Ohm
Power pins capacitors
Place near VCCIN
Place near VOUT3V3
5V/5.2B
VCCIO
C7
C8
C9
C10
4.7uF
0.1uF
4.7uF
0.1uF
GND
GND
FTDI USB to UART
photo-stage
2022-05-17 9:34 PM
Sheet: 2/6


1
2
3
4
5
6
A
B
C
D
SERVO_0/1.1B
SERVO_1/1.1C
SERVO_2/1.1C
SERVO_3/1.1B
SERVO_4/1.1B
SERVO_5/1.1B
SERVO_6/1.1B
SERVO_7/1.1B
JP1
JP2
+6V/5.6C
+3V/5.6C
photo-stage
2022-05-17 9:34 PM
Sheet: 3/6

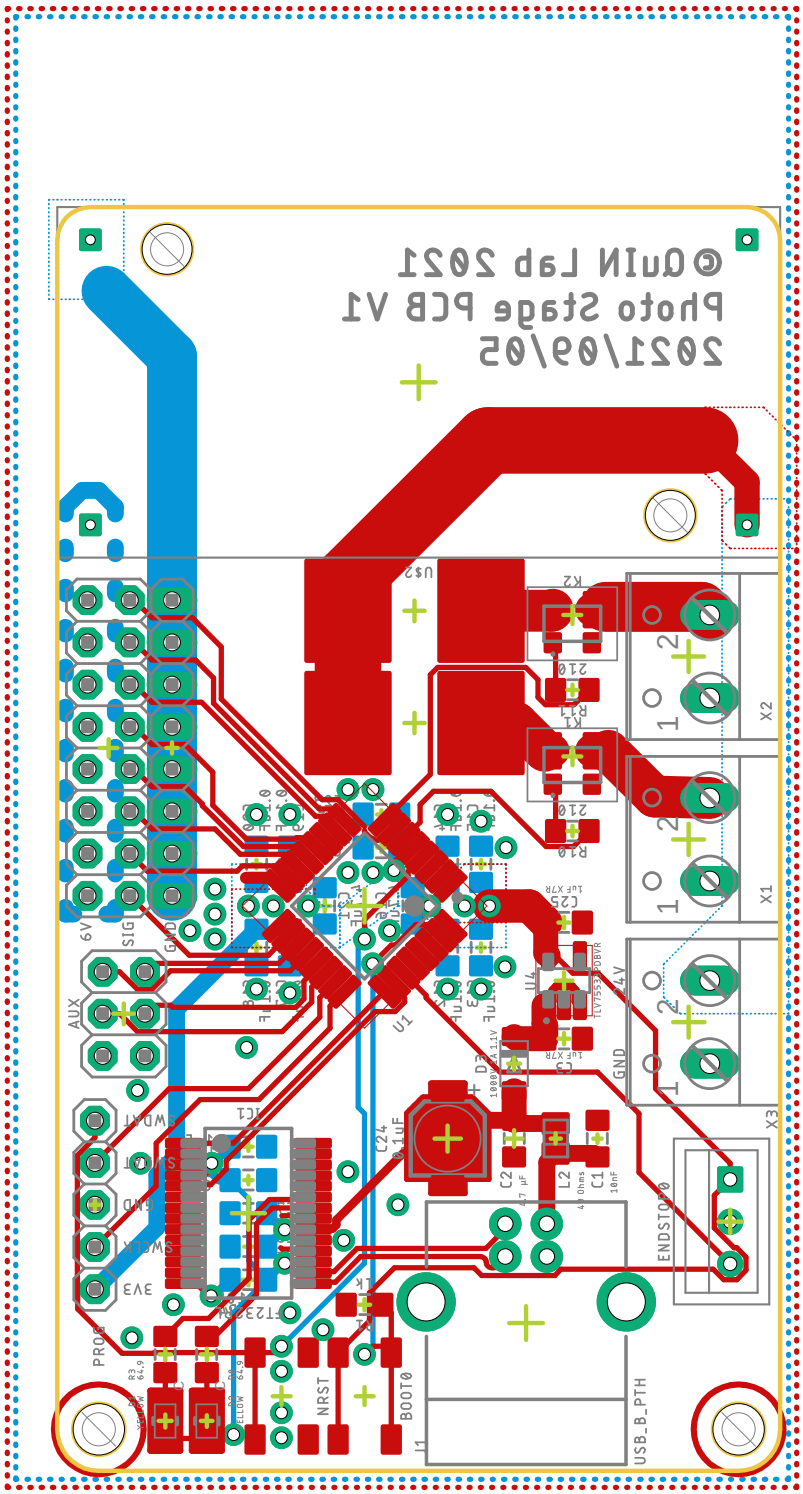

1
2
3
4
5
6
A
B
C
D
03-JST-B3B-XH-A
3V3/1.5B
ENDSTOP_0/1.1C
ENDSTOP0-1
ENDSTOP0-2
ENDSTOP0-3
GND
photo-stage
2022-05-17 9:34 PM
Sheet: 4/6


USB IN
USB_B_PTH
USB
GND
D+
D-
VBUS
SHLD1*2
J1
DP/2.2B
DM/2.2B
VBUS
GND
GND
L2
40 Ohms
VBUS
5V/2.5C
C1
10nF
C2
0.1uF
C24
4.7 µF
GND
GND
GND
24V IN
X3-2
X3-1
24V/6.4B
GND
3V3 Regulator
U4
D3
5V/2.5C
1000V 1A 1.1V
EN
IN
OUT
GND
3
1
5
2
3V3/4.3C
C3
1uF X7R
C25
1uF X7R
TLV75533PDBVR
GND
GND
GND
6V Regulator
U$2
24V/6.4B
IN+
IN-
OUT+
OUT-
+6V/3.3B
-6V/3.3B
GND
YIZHET_LM2596_DCDC
Power
photo-stage
2022-05-17 9:34 PM
Sheet: 5/6


Forward current: I_F = 10mA
Forward voltage @ 10mA: V_F = 1.2V
Logic voltage: V_IO = 3.3V
Series resistance: R_S = (V_IO - V_F) / I_F = 210Ohm
RELAY_0_SIGNAL/1.1B
R10
210
K1
TLP3480
F1
F1'
4A
24V/5.0C
X1-2
X1-1
GND
GND
RELAY_1_SIGNAL/1.1B
R11
210
K2
TLP3480
F2
F2'
4A
24V/5.0C
X2-2
X2-1
GND
GND
Relays
photo-stage
2022-05-17 9:34 PM
Sheet: 6/6

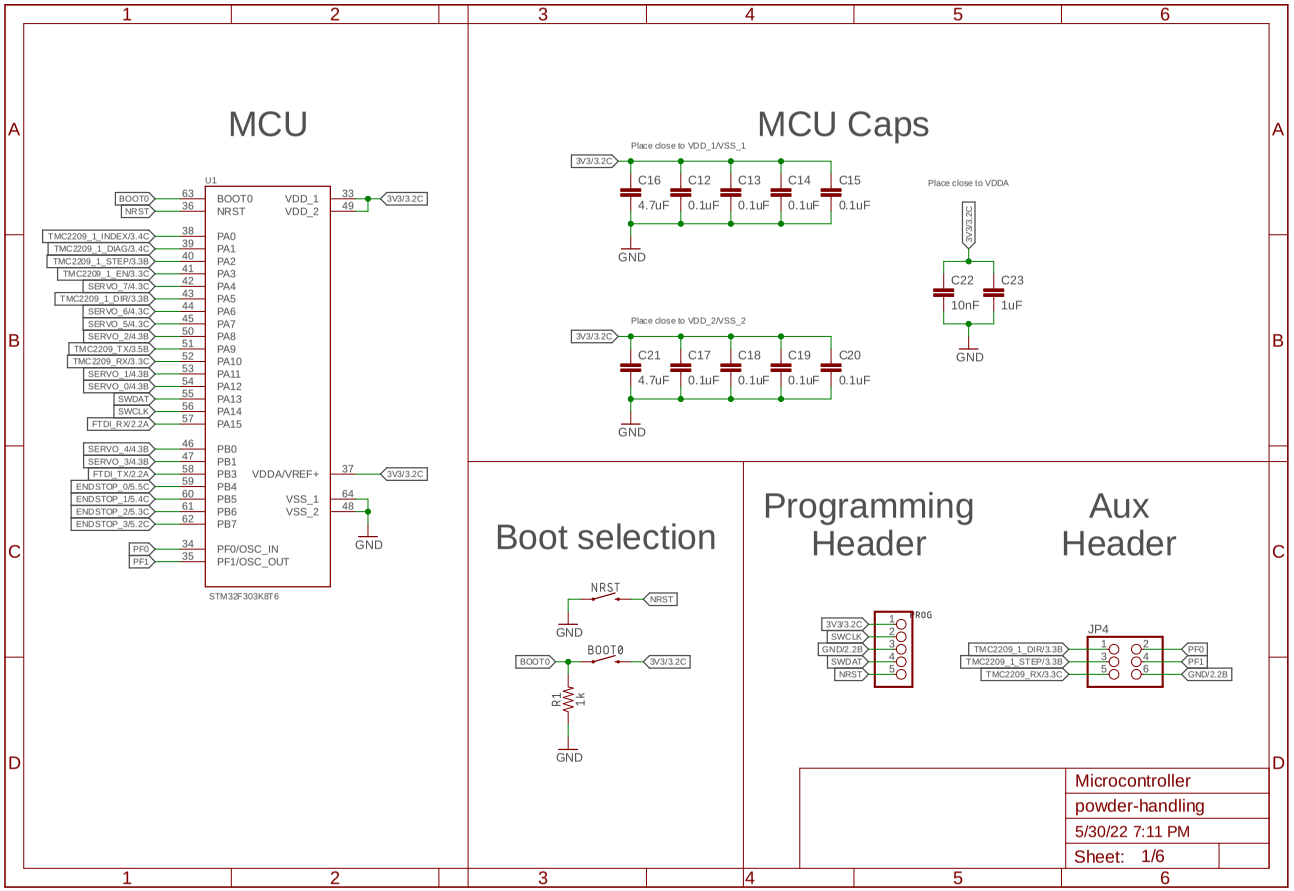
Appendix H

Board of the *Photo Stage* PCB in *Autodesk Eagle*


©QuIN Lab 2021
Photo Stage PCB V1
2021/09/05
X1
X2
X3
GND
4.4V
ENDSTOP0
USB_B_PTH
NRST
BOOT0
PROG
3V3
AUX
6V
SIG
GND
SWDAT
SWCLK
U1
U4
U5
J1
C24
0.1uF
C2
4.7 µF
L2
4.0mm
C1
10nF
C25
1uF X7R
C3
1uF X7R
D3
1000V 1.1V
R10
210
R11
210
K1
K2

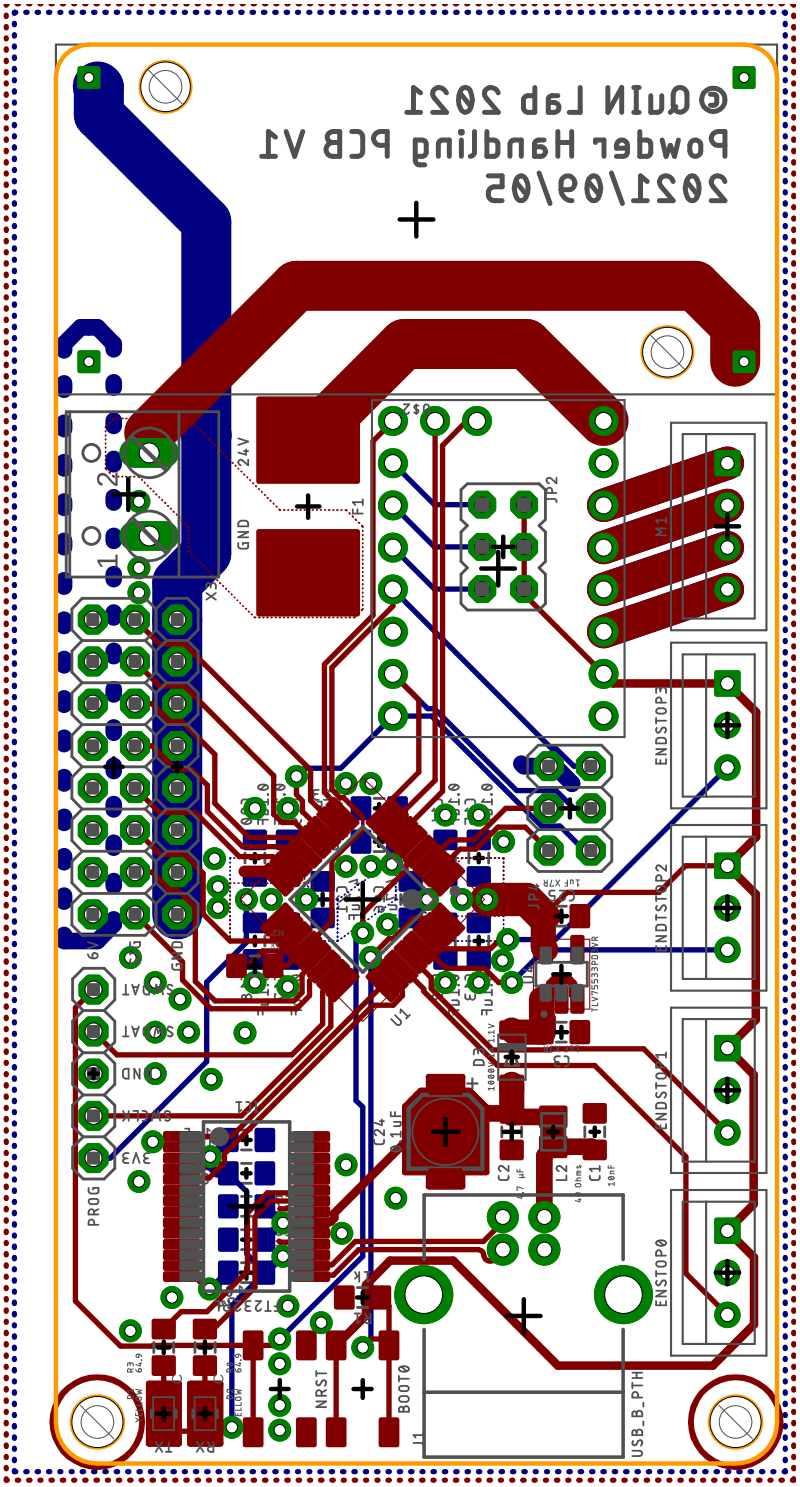
Appendix I

Schematic of the *Powder Handling Sliding Dispenser* PCB in *Autodesk Eagle*


MCU
U1
BOOT0
NRST
VDD_1
VDD_2
3V3/3.2C
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB0
PB1
PB3
PB4
PB5
PB6
PB7
VDDA/VREF+
VSS_1
VSS_2
GND
PF0/OSC_IN
PF1/OSC_OUT
STM32F303K8T6
TMC2209_1_INDEX/3.4C
TMC2209_1_DIAG/3.4C
TMC2209_1_STEP/3.3B
TMC2209_1_EN/3.3C
SERVO_7/4.3C
TMC2209_1_DIR/3.3B
SERVO_6/4.3C
SERVO_5/4.3C
SERVO_2/4.3B
TMC2209_TX/3.5B
TMC2209_RX/3.3C
SERVO_1/4.3B
SERVO_0/4.3B
SWDAT
SWCLK
FTDI_RX/2.2A
SERVO_4/4.3B
SERVO_3/4.3B
FTDI_TX/2.2A
ENDSTOP_0/5.5C
ENDSTOP_1/5.4C
ENDSTOP_2/5.3C
ENDSTOP_3/5.2C
MCU Caps
Place close to VDD_1/VSS_1
C16
C12
C13
C14
C15
4.7uF
0.1uF
Place close to VDD_2/VSS_2
C21
C17
C18
C19
C20
Place close to VDDA
C22
C23
10nF
1uF
Boot selection
NRST
BOOT0
R1
1k
Programming Header
PROG
JP4
Aux Header
TMC2209_1_DIR/3.3B
TMC2209_1_STEP/3.3B
TMC2209_RX/3.3C
PF0
PF1
GND/2.2B
Microcontroller
powder-handling
5/30/22 7:11 PM
Sheet: 1/6

Appendix J

Board of the *Powder Handling Sliding Dispenser* PCB in *Autodesk Eagle*


©QuIN Lab 2021
Powder Handling PCB V1
2021/09/05
24V
GND
F1
JP2
M1
ENDSTOP3
ENDSTOP2
ENDSTOP1
ENSTOP0
USB_B_PTH
U1
U2
C24
4.7UF
C2
C1
L2
D3
NRST
BOOT0
PROG
3V3
GND
SWDAT
SWCLK
RX
TX