



Steiner, Sebastian (2018) *From code to molecule: a versatile, modular, lab-scale automation strategy and platform for organic synthesis*. PhD thesis.

<https://theses.gla.ac.uk/30904/>

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses

<https://theses.gla.ac.uk/>
research-enlighten@glasgow.ac.uk

**From Code to Molecule:
A Versatile, Modular, Lab-Scale Automation
Strategy and Platform for Organic Synthesis**



University
of Glasgow

Sebastian Steiner

A thesis submitted to the University of Glasgow

for the degree of Doctor of Philosophy

School of Chemistry

College of Science and Engineering

October 2018

*The scientist merely explores that which exists,
while the engineer creates what has never existed before.*

Attributed to both Theodore von Kármán and Albert Einstein

Acknowledgements

This project was carried out between May 2015 and October 2018 in the Department of Chemistry at the University of Glasgow. Many people helped and advised me over this period and I really appreciate it. In particular, I would like to thank:

Prof. Lee Cronin for giving me the opportunity to work in his group and for his support, encouragement and supervision during these years.

My mentors in the group, **Dr. Anna Andreou** and **Dr. Stefan Glatzel**, who first introduced me into the project and helped the universal synthesiser become reality. Their work on the diphenhydramine hydrochloride chemistry, the platform architecture, and the early software solutions provided me with a solid foundation to build on.

My team and subgroup leaders along the way, **Dr. Geoffrey Cooper**, **Dr. Andrew Surman**, **Dr. Philip Kitson**, **Dr. Alon Henson** and **Dr. Jarosław Granda**. Thank you for your help, your guidance, and your advice.

My colleagues and predecessors on the project:

Dr. Trevor Hinkley for his work on the old device firmware. **Dr. Jon Trinder** and **Francis Jamieson** for their help with the new device firmware. **Dr. Gerardo Aragon-Camarasa** for developing ChemOS. Even though it was eventually superseded, it helped me avoid many pitfalls. **Graham Keenan** for his help in developing the ChemOS's successor, the Chempiler, as well as for his good humour and for recommending some amazing games. **Dr. Jarosław Granda** for his work on the rufinamide synthesis, and his work on image recognition for the liquid/liquid separator. **Dr. Davide Angelone** for carrying on the project. In particular, I would like to thank my student, **Jakob Wolf**, for his work on the sildenafil synthesis. His enthusiasm, work ethics, and his keen intellect were invaluablely helpful.

Our glassblower, **John Liddell**, for the marvellous pieces of bespoke glassware he crafted so expertly.

Our dedicated technical and administrative staff, **Jim McIver**, **Dr. Diana Castro**, **Dr. Emma Carrick**, **Amanda McGarvey**, and soon-to-be-doctor **Stuart Marshall**, for helping me with everything and anything, and in many cases going well above and beyond the call of duty. Without you all, the group would probably collapse spectacularly in about two and a half minutes.

My colleagues and friends in the Cronin Group, if not for keeping me sane, then at least for tolerating my lunacy. Especially **Vasilis**, who dragged me to the cinema every now and then to subject me to human contact; **David**, who went climbing with me despite of being afraid of heights; **James** and **Christoph** for hitting me in the head with swords (don't worry, I hit them back); **Sergey**, for many evenings spent fixing things that weren't broken, improving things that worked just fine, and obsessively organising labs that were doomed to entropic decay, and also for only occasionally treating me like the idiot I am; **Danny**, for many interesting conversations, relentless banter, and generally being a friend. Obviously, the list goes on and on and the thesis is long enough as it is, so if you're not in this list, this does not mean I am not happy to have spent the time with you!

My family, in particular my parents **Gerhard** and **Gudrun** and especially my love **Pia**, for always supporting me. I don't think I would have made it through in one piece without you, thank you for your patience and your understanding and for always being there when I need you.

ABSTRACT

The work presented in this thesis focused on the automation of multistep organic syntheses on a laboratory scale in batch. Automation has received much attention in the chemical sciences and many automated synthesis solutions are commercially available or in development, yet all those solutions are focused on narrow aspects of the wider problem of automating chemical synthesis. In particular, a gap in the currently employed technologies was identified where synthesis in batch on a gram scale was concerned. Furthermore, many available solutions are monolithic and cannot easily be adapted for new applications, partially due to shortcomings of the hardware design, and partially due to the bespoke software controlling them.

To address this need for a new approach to automated synthesis, a novel strategy comprising hardware modules dedicated to individual unit operations (as opposed to other solutions built around specific chemical transformations and a modular flexible control software was developed. To enable the hardware development, liquid handling hardware was built and optimised, and four major modules for the unit operations of mixing under heating or cooling, liquid/liquid separation, filtration, and evaporation as well as several auxiliary modules were developed and tested.

The control software orchestrating the operation of the synthesis platform was modelled after a compiler in modern computer science, separating the synthetic operations from the physical hardware of the platform. This way, synthetic procedures can be transferred between different platforms, and new hardware modules can be added to the system at will. To enable the average synthetic chemist to use the system, a rudimentary scripting language for chemical operations was developed.

To prove the capabilities of the platform, three Active Pharmaceutical Ingredients (APIs) were synthesised in a fully automated fashion in yields and purities comparable to those obtained by hand. The automated reactions included a Grignard reaction and a chlorosulfonation, to name but a few. Additionally, the synthesis of one of the APIs was repeated on two physically different platforms simply by executing the same code on both systems.

ABSTRACT

ABBREVIATIONS

ADC	Analog-Digital Converter
API	Application Programming Interface (in computer sciences), or Active Pharmaceutical Ingredient (in chemistry)
ASCII	American Standard Code for Information Interchange
ASF	Atmel Software Framework
AVR	A microcontroller architecture. There is no official consensus about what the acronym stands for.
CAD	Computer-aided Design
CAN	Controller Area Network
CPU	Central Processing Unit
DCM	Dichloromethane
DHCP	Dynamic Host Configuration Protocol
DMF	<i>N,N</i> -dimethylformamide
EEPROM	Electrically Erasable Programmable Read-Only Memory
FDM	Fused Deposition Modelling
GPIO	General Purpose Input/Output
I.D.	Internal Diameter
IP	Internet Protocol
ISR	Interrupt Service Routine
IV	intravenous
LED	Light Emitting Diode
MAC	Media Access Control address

ABBREVIATIONS

MCU	Microcontroller Unit
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
O.D.	Outer (external) Diameter
OSC	Open Sound Control
PCB	Printed Circuit Board
PEEK	Polyether ether ketone
PLA	Polylactic acid
PMIC	Programmable Multilevel Interrupt Controller
PoE	Power over Ethernet
PP	Polypropylene
PVC	Polyvinyl chloride
PVDF	Polyvinylidene fluoride
PWM	Pulse Width Modulation
RS232	Recommended Standard 232, a serial communication standard
RS485	Recommended Standard 485, a serial communication standard
SAM	Smart ARM-based microcontroller. ARM stands for Advanced RISC Machine, where RISC stands for Reduced Instruction Set Computer.
SMD	Surface Mounted Device
SPI	Serial Peripheral Interface
STL	Stereolithography file (a file format)
TCP	Transmission Control Protocol
THT	Through Hole Technology
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Data Protocol

ABBREVIATIONS

UHDPE Ultra-high density polyethylene

USB Universal Serial Bus

ABBREVIATIONS

TABLE OF CONTENTS

ABSTRACT	I
ABBREVIATIONS	III
TABLE OF CONTENTS	VII
INTRODUCTION	1
1 Iterative Synthesis	1
2 Robotic Platforms	9
3 Flow Chemistry	13
4 Automated Batch Reactors	19
5 The Chemputer Concept	25
AIMS	28
RESULTS AND DISCUSSION	29
6 Hardware Development	29
6.1 Pumps and Valves	29
6.1.1 Prior State of the Art	29
6.1.2 Transfer to Autodesk Inventor and Initial Improvement Efforts	35
6.1.3 Improving the Alignment.....	36
6.1.4 The Evolution of the Hall Effect Sensor	42
6.1.5 Replacement of the Pump Motor	49
6.1.6 Improvements to Usability and Aesthetics	51
6.1.7 Project Documentation	54
6.2 Chemputer Setup.....	57
6.2.1 Reactor Module	57
6.2.2 Evaporation Module	60
6.2.3 Automated Liquid/Liquid Extractor (ALLEX).....	64

TABLE OF CONTENTS

6.2.4	Filtration Module	75
6.2.5	Inert Gas System	79
6.2.6	Reagent Storage	80
6.2.7	Other Modules	81
7	Software Development.....	87
7.1	Pump and Valve Firmware	87
7.1.1	Existing Hardware Specifications and Firmware Requirements	87
7.1.2	General Operation and Requirements.....	89
7.1.3	MCU Initialisation and General Setup.....	90
7.1.4	Ethernet Communication	93
7.1.5	The Command Mapper and the Formatted Network Print Utility ..	98
7.1.6	Stepper Motor Control.....	100
7.1.7	Pump Positioning Algorithms.....	109
7.1.8	Valve Positioning Algorithms	115
7.1.9	Device Configuration and Errors	122
7.1.10	Python API	123
7.1.11	Firmware Testing	123
7.2	The SerialLabware Project	125
7.3	The Chempiler.....	131
7.3.1	Motivation and Requirements	131
7.3.2	The ChemOS and “The Script”	132
7.3.3	Drafting a Specification for the Chempiler.....	134
7.3.4	Building the Chempiler.....	136
7.3.5	Moving Liquids	138
7.3.6	From XML to GraphML.....	142
7.3.7	The Chemical Assembly Language ChASM.....	147

TABLE OF CONTENTS

7.3.8	Additional Modules and Final Structure	150
7.3.9	Translation of a Synthetic Procedure into ChASM.....	154
8	Automated Syntheses.....	169
8.1	Synthesis of Diphenhydramine Hydrochloride (1)	170
8.1.1	Prerequisites and Initial Work	170
8.1.2	The Journey Begins with a Bromination.....	171
8.1.3	Pressing on Towards the Williamson Ether Synthesis	175
8.1.4	Cresting the First Summit with the Grignard Reaction	177
8.1.5	Commencing the End Game with the Hydrochloride Precipitation 180	180
8.1.6	Tying It All Together	181
8.1.7	Rebuilding the Platform and Starting Anew.....	183
8.1.8	Adding Purification and Cleaning	190
8.2	Synthesis of Rufinamide (2)	195
8.2.1	Prerequisites and Initial Work	195
8.2.2	Automation on the Small Platform	195
8.2.3	Transferring the Code to Another Platform	196
8.3	Synthesis of Sildenafil (3).....	199
8.3.1	Prerequisites and Initial Work	199
8.3.2	Manual Replication of the Reported Synthesis.....	200
8.3.3	Subjecting the Platform to Chlorosulfonic Acid	204
8.3.4	Initiating the Crystallisation of the Sulfonamide.....	206
8.3.5	Forming the Acid Chloride and Performing the Amide Coupling...207	207
8.3.6	Closing the Cycle with Potassium <i>tert</i> -Butoxide	208
8.3.7	Going the Distance: Running the Full Sequence	209
	CONCLUSIONS AND FUTURE WORK	211

TABLE OF CONTENTS

EXPERIMENTAL.....	217
9 Chemicals and Instrumentation	217
10 Computer Controlled Instrumentation.....	218
10.1 Reactor Module	218
10.2 Automated Liquid/Liquid Extraction Module	220
10.3 Solvent evaporation module	222
10.4 Filtration Module	223
10.5 Inert Gas System	225
10.6 Reagent Storage System	226
11 Synthesis and Characterisation of Compounds.....	229
11.1 Diphenhydramine hydrochloride.....	229
11.1.1 Diphenylmethanol (4).....	229
11.1.2 Bromodiphenylmethane (5)	231
11.1.3 2-(diphenylmethoxy)- <i>N,N</i> -dimethylethanamine (diphenhydramine) (6).....	232
11.1.4 2-(diphenylmethoxy)- <i>N,N</i> -dimethylethanamine hydrochloride (1) 234	
11.2 Rufinamide	236
11.2.1 1-[(2,6-Difluorophenyl)methyl]-1 <i>H</i> -1,2,3-triazole-4-carboxamide (rufinamide) (2).....	236
11.3 Sildenafil.....	238
11.3.1 5-Chlorosulfonyl-2-ethoxybenzoic acid (9)	238
11.3.2 2-Ethoxy-5-(4-methyl-1-piperazinesulfonyl)benzoic Acid (10) ..	240
11.3.3 4-[2-Ethoxy-5-(4-methyl-1-piperazinylsulfonyl)benzamido]-1- methyl-3-propyl-1 <i>H</i> -pyrazole-5-carboxamide (12).....	242

TABLE OF CONTENTS

11.3.4 1-[4-Ethoxy-3-(6,7-dihydro-1-methyl-7-oxo-3-propyl-1H-pyrazolo[4,3-d]pyrimidin-5-yl)phenylsulfoyl]-4-methylpiperazine (Sildenafil) (3) 244

REFERENCES 247

APPENDIX 257

I NMR Spectra 257

I.I Diphenhydramine Hydrochloride (1).....258

I.II 1-[(2,6-Difluorophenyl)methyl]-1*H*-1,2,3-triazole-4-carboxamide (rufinamide) (2)264

I.III 1-[4-Ethoxy-3-(6,7-dihydro-1-methyl-7-oxo-3-propyl-1H-pyrazolo[4,3-d]pyrimidin-5-yl)phenylsulfoyl]-4-methylpiperazine (Sildenafil) (3)268

II Engineering Drawings 273

III Control PCB Schematics 319

IV ChASM reference..... 327

IV.I.I Pumps and Valves.....327

IV.I.II Stirrer Plates and Overhead Stirrers329

IV.I.III Rotary Evaporator330

IV.I.IV Vacuum Pump332

IV.I.V Recirculation Chiller333

IV.I.VI Camera335

IV.I.VII Other335

INTRODUCTION

Organic synthesis is arguably one of the most labour-intensive branches of chemistry, requiring an expert chemist to manually execute a multitude of relatively simple unit operations such as mixing of chemicals, liquid-liquid extractions or filtrations. While automated approaches for some of those unit operations exist,¹ unified strategies tying together individual technologies are underdeveloped.

Currently available synthesis automation strategies can be grouped into three key technologies: iterative synthesis, robotic platforms and flow chemistry. A potential fourth technology, the automation of laboratory scale batch synthesis, has been investigated in the past, but is generally underrepresented in the modern literature. The following pages explore those technologies and their respective advantages and drawbacks. Automation efforts in disciplines other than bench-scale synthesis, such as analysis, biology, or plant-scale production, are not taken into account, as they do not apply to the challenge at hand. It must be noted that laboratory automation in some of those areas, most notably in the area of chemical and biochemical analysis, has progressed much further than in synthesis, with fully automated high-throughput workflows being the state of the art.

1 ITERATIVE SYNTHESIS

Iterative synthesis type approaches can be used for the sequence-controlled synthesis of oligo- and polymers. They employ solid support matrices to immobilise the product and elongate the chain by repeatedly coupling protected monomers in an iterative fashion. Examples of currently used iterative strategies include synthesis of peptides,² oligonucleotides,³ or oligosaccharides⁴ as well as iterative cross coupling.⁵

The first example of a fully automated iterative synthesiser (Figure 1) was reported by Merrifield in 1965.² The ingenious idea that led to this development was the concept of solid phase synthesis Merrifield published just two years earlier,⁶ which utilises an insoluble resin as support for the growing peptide chain. A carboxybenzyl

INTRODUCTION

(Cbz) protected amino acid is coupled to the resin, the amino group is deprotected, and another Cbz protected amino acid is coupled to the free amine using *N,N'*-dicyclohexylcarbodiimide (DCC). This procedure is repeated in an iterative fashion until the desired chain length is achieved, and the target peptide is eventually cleaved from the resin support. The pivotal advantage of the resin support is that the intermediate is immobilised on the solid resin beads and can therefore be purified by simple filtration. Monomers and coupling agents can be employed in large excess, driving the reaction to completion, and then be washed away.

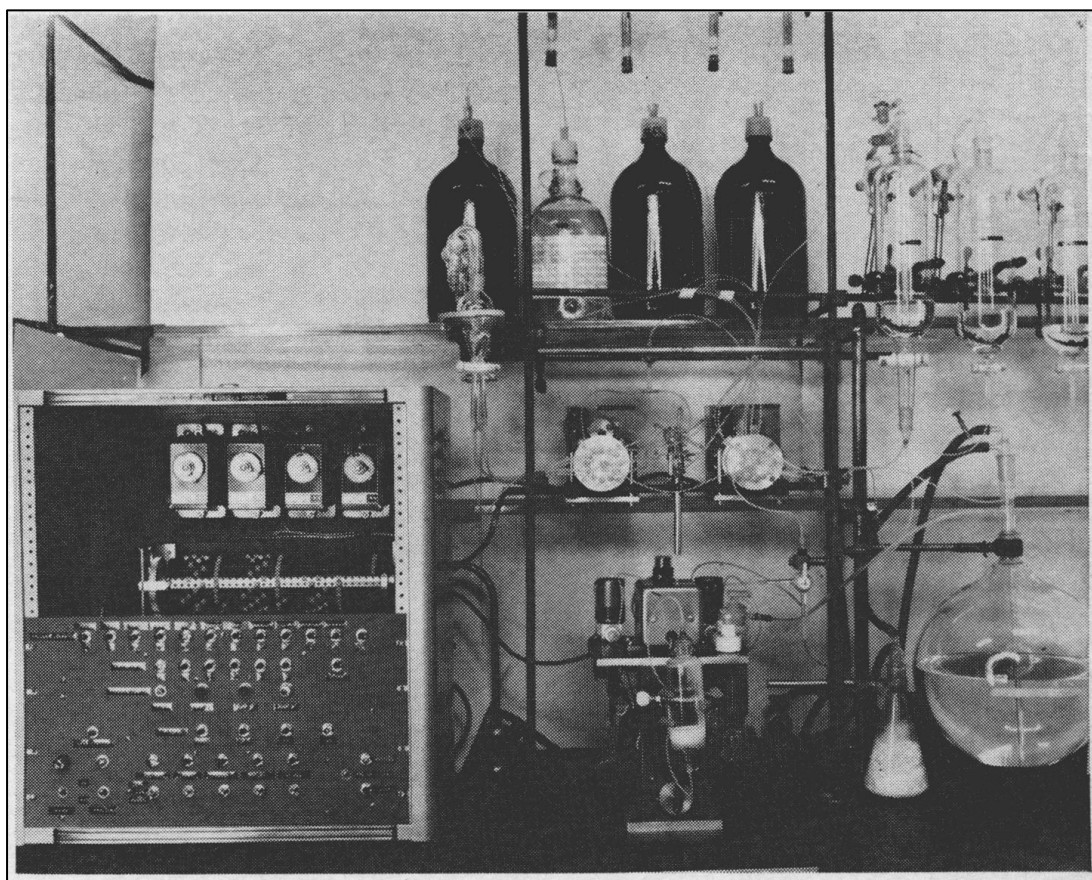


Figure 1: Photo of the first automated peptide synthesiser built by Merrifield. From Merrifield, R. B., Automated synthesis of peptides. *Science* **1965**, *150* (3693), 178-85. Reprinted with permission from AAAS.

This first automated peptide synthesiser completely transformed the discipline of protein synthesis. Around the same time Merrifield reported his solid phase method, Zahn⁷, Katsyannis⁸ and the Shanghai Institutes of Biochemistry and Organic Chemistry⁹ reported the synthesis of insulin, a protein consisting of 51 amino acids,

INTRODUCTION

a colossal undertaking at the time. To get a sense of the scale of the project, Katsoyannis published over a dozen papers about the project between 1961¹⁰ and 1966¹¹, and Zahn's 1963 paper⁷ lists 10 authors. Using his new automated peptide synthesiser, Merrifield and his colleague Bernd Gutte published the synthesis of ribonuclease A, a 124-amino acid protein, in 1969¹² after just one year of development, according to Merrifield.¹³

Nowadays, fifty years later, automated peptide synthesisers are a staple of modern biochemistry. Numerous models are commercially available (Figure 2), as are various different resins as well as appropriately protected amino acids, and the protecting group chemistry and coupling agents have progressed immensely since Merrifield's days.¹⁴



Figure 2: The Biotage® Initiator+ Alstra™ is one example of the various commercially available peptide synthesisers at the time of writing (image reproduced with permission of Biotage AB).

The next class of molecules that was brought to heel by iterative synthesisers were oligonucleotides, that is, DNA and RNA. The first successful synthesis of a dinucleotide in the correct linkage was reported by Michelson and Todd in 1955.¹⁵ Over the following decade the coupling chemistry was greatly improved,¹⁶⁻¹⁸ and following Merrifield's work in the mid-1960s, solid supported oligonucleotide synthesis became the obvious next step. This development was enabled by two

fundamental discoveries. First, Letsinger and co-workers developed the phosphite method,¹⁹ a coupling chemistry which provided short enough reaction times and high enough yields to make large-scale oligonucleotide synthesis feasible. This was followed by Ogilvie and Nemer²⁰ as well as Matteucci and Caruthers²¹ independently pioneering the use of silica as a suitable solid support for nucleotide chemistry. Based on those two key technologies, a fully automated DNA synthesiser was developed by Ogilvie in 1981.³ Figure 3 shows the chemistry used in their synthesiser.

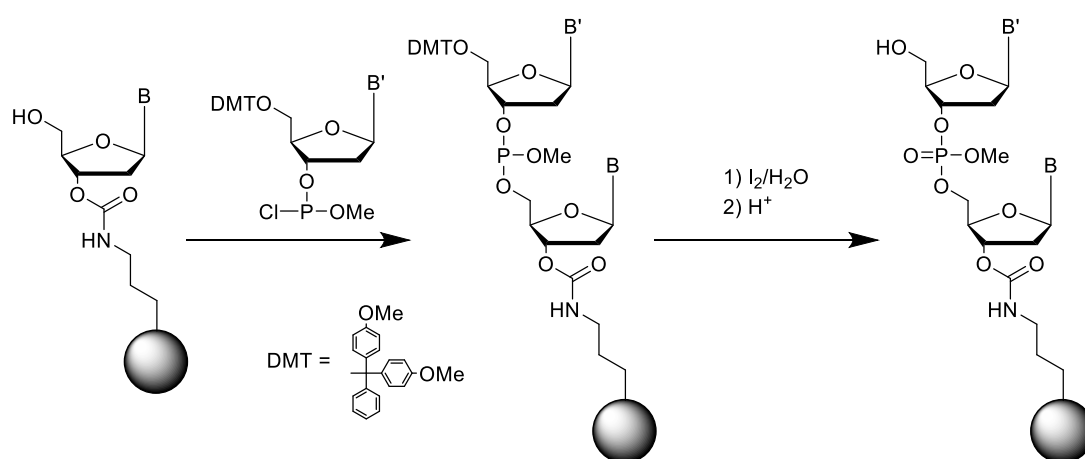


Figure 3: Solid supported DNA synthesis using the phosphite ester method.³ B and B' stand for thymine, *N*-benzoylcytosine, *N*-benzoyladenine, or *N*-isobutyrylguanine.

Since then, the instrumentation and methodology have been refined and diversified. By now, full artificial genes upwards of 10,000 bases are synthetically accessible. Custom “oligos”, as shorter oligonucleotides in the range of dozens to hundreds of bases are colloquially referred to, can be inexpensively purchased from a number of vendors. For further reading on the subject, Kosuri and Church²² give an excellent overview of current technologies and applications in their 2014 review.

Automated oligonucleotide synthesis is now a cornerstone of modern molecular biology. Many fundamental methods such as PCR would not be possible without fast and cheap access to custom DNA and RNA sequences, so the importance of this automation technology can hardly be understated.

Twenty years after Ogilvie's DNA synthesiser, the Seeberger group successfully automated the iterative synthesis of the next class of biomolecules: the

INTRODUCTION

oligosaccharides.⁴ In his 2005 review²³ on the subject, Seeberger explains the difficulty of automating oligosaccharide synthesis. First, oligosaccharides are not necessarily monotonic linear molecules. Every sugar monomer has a number of attachment sites which requires sophisticated orthogonal protecting group strategies to ensure chemoselective assembly. Second, the glycosidic bond itself can take one of two isomeric forms and therefore stereoselective coupling strategies are needed.



Figure 4: First automated oligosaccharide synthesiser.²³ Reprinted by permission from Springer Nature: Nature Reviews Drug Discovery Automated synthesis of oligosaccharides as a basis for drug discovery, Seeberger & Werz, 2005

Although the first solid phase synthesis of oligosaccharides was reported as early as 1971²⁴ and the employed methodologies were refined in the following decades,²⁵⁻²⁷ it wasn't until 1998 that work towards fully automating the synthesis began.²⁸

INTRODUCTION

Seeberger and his group modified a commercially available peptide synthesiser (Figure 4), selected appropriate monomers and coupling strategies, and eventually published the first fully automated synthesis of several oligosaccharides, including a branched dodecamer three years later.⁴

The technology has matured in the meantime, with the automated assembly of a 50mer reported in 2017.²⁹ A spin-off company named GlycoUniverse was founded by Seeberger in 2013 and, at the time of writing, it is marketing its first commercial oligosaccharide synthesiser, the GLYCONEER® (Figure 5).



Figure 5: The GLYCONEER® automated oligosaccharide synthesiser (image reproduced with permission of GlycoUniverse GmbH & CO KGaA).

Iterative strategies can also be employed for the synthesis of small molecules. Recently, the Burke group demonstrated the use of iterative cross coupling of MIDA boronates for the synthesis of a range of natural products (Figure 6).³⁰ This technique involves the use of building blocks functionalised with a halogen and a boronic acid moiety protected with *N*-methyliminodiacetic acid (MIDA). Those building blocks can undergo Suzuki-Miyaura cross-coupling with an unprotected organoboronate without the risk of uncontrolled oligomerisation. The coupling product can be purified followed by removal of the MIDA protecting group using aqueous sodium hydroxide, and another B-protected building block can be attached. Encouraged by those findings, Burke *et al.* conducted a systematic review of a large number of structures of known polyene natural products and claimed that 75% of those

products could be made from just 12 building blocks.³¹ While this may be a somewhat sensationalist exaggeration (*vide infra*), it is probably safe to say that a large number of different molecules could feasibly be assembled from a small number of building blocks using their methodology.

When setting out to automate their method⁵ in a similar way as peptide or DNA synthesis, they encountered a problem. Solid phase synthesis requires a common handle to attach the first monomer to the solid support, but the range of natural products Burke *et al.* were targeting lacked such a handle. Instead, they made a fortuitous discovery: MIDA boronates do not elute from silica when methanol and diethyl ether are used as eluents, virtually regardless of the organic moiety attached to it. However, THF would readily elute them. Thus, a catch and release type of purification could be employed, paving the way towards fully automating the assembly of small molecules using the iterative cross coupling methodology.



Figure 6: Burke's iterative cross coupling machine. From Li, J.; Ballmer, S. G.; Gillis, E. P.; Fujii, S.; Schmidt, M. J.; Palazzolo, A. M.; Lehmann, J. W.; Morehouse, G. F.; Burke, M. D., Synthesis of many different types of organic small molecules using one automated process. *Science* **2015**, 347 (6227), 1221-6. Reprinted with permission from AAAS.

All the aforementioned technologies have two things in common. First, they are easily automated. The handling operations required are limited to pumping reagent solutions and performing filtrations. Also, the chemistries employed are limited to only a handful of transformations per cycle, which are repeated over and over to

assemble virtually any target molecule of the respective class. The reagents and conditions employed can thus be tuned and optimised to a high degree. In fact, the key technologies enabling each of the aforementioned synthesisers were not so much the required instrumentation, but rather the chemical strategies for support, coupling, and deprotection.

Second, all those iterative syntheses are fundamentally limited to one class of molecules. Martin Burke disagrees with this assessment to a certain degree, claiming that iterative strategies are the key to a large fraction of chemical space.³² The bold claim of synthesising most polyene natural product motifs with just 12 building blocks and one coupling reaction³¹ diminishes somewhat when the qualifications “polyene” and “motifs” are taken into account. The paper started from a database of natural products containing 238,541 entries at the time and extracted a subset of 2,839 or 1.2% containing three or more conjugated double bonds, none of which are contained in a ring of less than 12 members. They then removed the two olefinic termini, and retrosynthesised the remaining substructure, arriving at 12 building blocks required to assemble more than 75% of those substructures. However, to finish the total synthesis, they reasoned that over 600 unique capping elements are required to synthesise 75% of the investigated target molecules. Moreover, they did not state the overlap between the 75% of the core motifs and the 75% of the capping elements. Thus, at best, over 600 unique building blocks are required to assemble less than 1% of natural product space. It can be argued that this does not satisfy the definition of universality.

In a recent review,³³ Lehmann, Blair and Burke identified a range of other iterative syntheses giving access to classes of natural products, albeit without mentioning useful, generalisable automation strategies for them. While it is certainly conceivable that current technologies can be broadened in scope, and new coupling chemistries may unlock new angles at assembling small molecules, it remains questionable if an iterative approach could ever be general enough to access an appreciable fraction of chemical space.

2 ROBOTIC PLATFORMS

The next common synthesis automation strategy is the use of robotic systems. In this context, robotic systems are characterised by being mostly driven by one or more mechanical actuators such as a robotic arm, or an XYZ Cartesian gantry. Using a robotic arm to replace a human chemist is a rather obvious thought, while the use of Cartesian robots is well suited for addressing arrays of vessels. Both benefit greatly from the large body of work done for industrial applications such as automotive manufacturing or CNC machining, providing tried-and-true machinery as a basis for their development.

One of the first applications of robotics in the chemistry laboratory was in analytical chemistry,³⁴ and to this day chemists will be most familiar with robots for analytical equipment in the form of autosamplers. Yet, robotic arms were investigated for the automation of compound synthesis as early as the mid-1980s (Figure 7),³⁵ but robotic platforms did not receive much attention until the 1990s.³⁶

In the following years, robotic workstations became a standard tool for screening compounds for biological activity in so-called high-throughput screenings or HTS.³⁷ This allowed the rapid screening of a large number of chemical entities against a variety of biological targets by using Cartesian liquid handlers to prepare assays in well plates and robotic arms to transfer the well plates between workstations. However, in those early days, the screening candidates were usually taken from already existing libraries,³⁸ thus shifting the bottleneck to synthesising more candidates in a time-efficient manner.

Enter combinatorial chemistry. As an umbrella term, combinatorial chemistry describes a range of techniques for the rapid preparation of large compound libraries by systematic or random combination of building blocks.³⁹ By now combinatorial chemistry constitutes its own branch of organic synthesis and a full review of the field exceeds the scope of this chapter. The important aspect with regards to this thesis is that due to its highly parallel nature, combinatorial chemistry lends itself well to robotic automation.⁴⁰⁻⁴¹

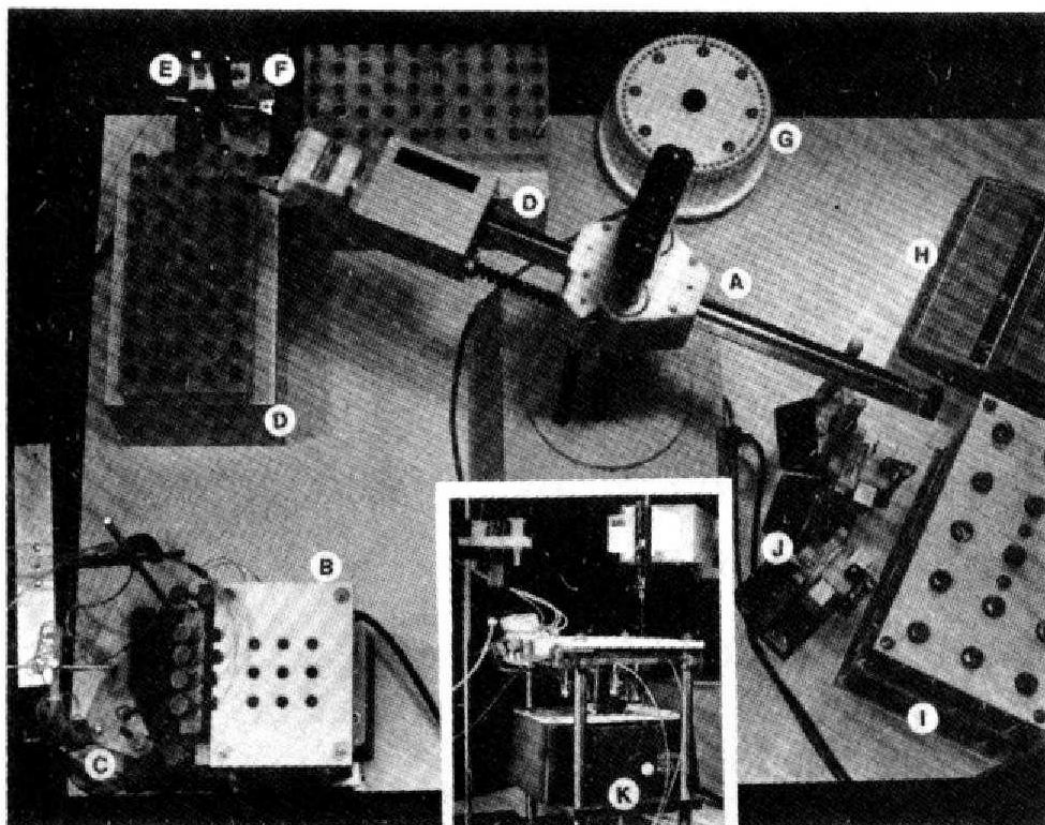


Figure 7: Synthesis robot used by Fuchs *et al.* to optimise an organic reaction in 1984.³⁵ A) Robotic arm, B) reactor station, C) solvent/gas/vacuum source, D) aliquot archive station, E) workup station, F) syringe and needle wash station, G) turntable, H) integrator, I) reagent station, J) parking station, K) front view. Reprinted (adapted) with permission from Frisbee, A. R.; Nantz, M. H.; Kramer, G. W.; Fuchs, P. L., Robotic orchestration of organic reactions: yield optimization via an automated system with operator-specified reaction sequences. *Journal of the American Chemical Society* **1984**, 106 (23), 7143-7145. Copyright 2018 American Chemical Society.

Combining those two technologies in their automated form essentially yields a fully automated medicinal chemistry laboratory. In fact, in 2013 researchers at Eli Lilly and Co. reported building a fully automated, remote-controlled robotic med-chem lab (Figure 8).⁴² Researchers could design experiments from the comfort of their desk without even having to be on site, and a collection of interconnected robotic workstations would perform compound synthesis, purification, characterisation, and biological testing and report the results.

INTRODUCTION

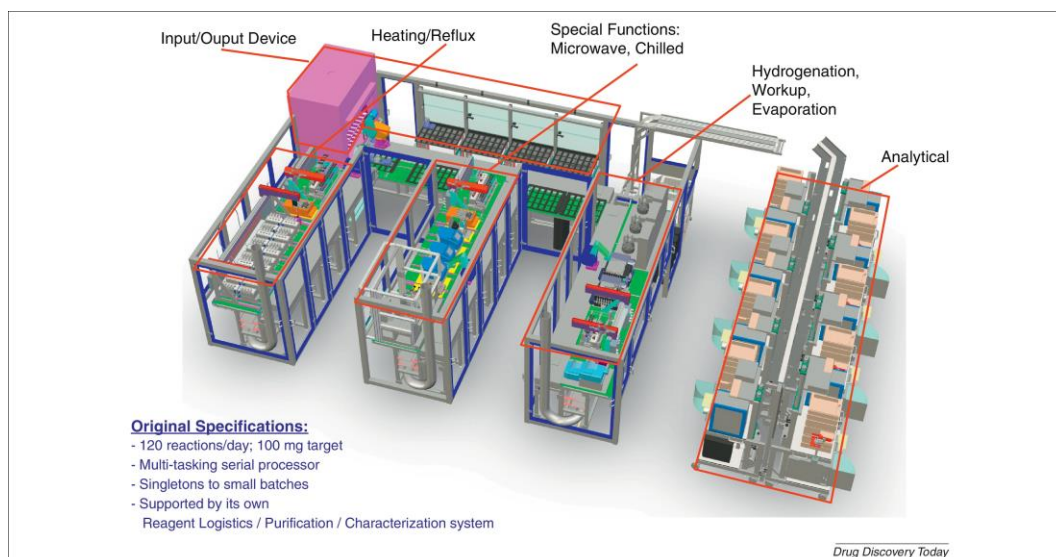


Figure 8: Eli Lilly's robotic med-chem lab.⁴² Reprinted from Godfrey, A. G.; Masquelin, T.; Hemmerle, H., A remote-controlled adaptive medchem lab: an innovative approach to enable drug discovery in the 21st Century. *Drug Discov Today* 2013, 18 (17-18), 795-802., with permission from Elsevier

Robotic workstations are also employed in other fields, such as process research and development⁴³ or materials science.⁴⁴ Many turnkey and customised solutions are commercially available,⁴⁵⁻⁴⁷ and synthesis robots are by now a common sight in the pharmaceutical industry.⁴⁸

For all their potential, robotic solutions also have their drawbacks. Multistep reactions are possible but often require specialised methodologies such as immobilised reagents⁴⁹⁻⁵⁰ or scavenger resins.⁵¹ Also, the amount of material obtained is usually quite small, in the milligram range. While being a boon in the early stages of the drug development process due to reduced reagent use and waste production, it can actually be an impediment in the later stages when material should be isolated or process development data should be obtained.⁵²

INTRODUCTION

3 FLOW CHEMISTRY

The third automation strategy commonly encountered today in chemistry is continuous flow. It is essentially a paradigm shift from the classical way chemistry is conducted in a flask in batch, to a continuous processing of materials in a roughly tubular reactor. This has a number of advantages over batch chemistry:⁵³ the short diffusion lengths inside a flow reactor ensure efficient mixing and therefore improved homogeneity. The high surface-to-volume ratio provides superior heat transfer into or out of the reaction mixture, allowing efficient heating, as well as safe handling of strongly exothermic reactions. The high wall thickness of most classical flow reactors in relation to the internal diameter allows reactions to be conducted at elevated pressures safely, without the need for cumbersome autoclaves. Additionally, at any given time, the reactor only contains miniscule amounts of reagents, thereby further improving chemical safety.⁵⁴

However, there are also limitations. One of the core problems of flow chemistry is its general incompatibility with flowing solids. Although considerable effort is aimed at handling particulates in flow,⁵⁵ precipitation inside the flow reactor still presents a major challenge. Another disadvantage is that reaction conditions obtained in batch can't always easily be transferred to flow without further optimisation.⁵⁶

The entire field of flow chemistry is vast. The interested reader is referred to the aptly named review "The Hitchhiker's Guide to Flow Chemistry" by Seeberger *et al.*⁵⁷ which gives an excellent account of the theory, techniques and applications of modern flow chemistry. The important consideration in the context of this work is the scope and limitations of flow chemistry, as the fundamental design choice to focus on automating batch processes is largely based on the belief that flow chemistry is a specialist tool for certain applications and not readily generalisable.

Many a review has been written on the question whether to perform syntheses in batch or in flow.^{53, 58-60} The generally accepted answer is: it depends. Flow chemistry is a powerful tool where exotic or dangerous reaction conditions are required. One such example is the catalytic hydrogenation. The handling of hydrogen gas always

poses a safety issue. A recent publication on the subject⁶¹ elegantly sums up the challenges of conducting laboratory scale hydrogenations in batch:

“

Fire, runaway reactions and explosions are commonly associated with hydrogenations due to the involvement of pyrophoric catalysts, hydrogen, flammable solvents and pressure.

“

The alternative, particularly for larger quantities, would be to use a device called the H-Cube marketed by ThalesNano (Figure 9).⁶²⁻⁶³ This device generates hydrogen on demand by means of water electrolysis, and safely conducts hydrogenations at elevated pressures and temperatures in flow, thus bypassing the aforementioned fires, runaway reactions and explosions.



Figure 9: The H-Cube by ThalesNano.⁶³ Reprinted from Dormán, G.; Kocsis, L.; Jones, R.; Darvas, F., A benchtop continuous flow reactor: A solution to the hazards posed by gas cylinder based hydrogenation. *Journal of Chemical Health and Safety* **2013**, 20 (4), 3-8, with permission from Elsevier.

Another case where flow chemistry can greatly improve safety is the use of diazomethane. Diazomethane is one of the most versatile C1 building blocks, and very useful due to its generally fast and clean reactions.⁶⁴⁻⁶⁵ However, the dangers associated with it are well appreciated: it is highly toxic,⁶⁶⁻⁶⁷ carcinogenic,⁶⁸ and spontaneously explosive.⁶⁹ Standard laboratory techniques⁷⁰ for reactions with

diazomethane generate it *in situ* either as gas, or preferably as solution in ether, and use it right away. Alternatively, reactions with diazomethane can be conducted in a dual channel⁷¹ or tube-in-tube⁷² flow reactor consisting of two flow paths separated by a gas permeable membrane. The diazomethane is generated in one flow path and permeates into the other flow path, where it reacts with the substrate. Effluents of both streams are directed into a quenching solution, destroying any unreacted diazomethane, thus limiting the amount of free diazomethane present at any one time. Similar setups have also been demonstrated for safely conducting other problematic liquid/gas reactions such as ozonolysis⁷³ or reactions with hydrogen cyanide.⁷⁴

The utility of flow chemistry is not limited to the safe use of hazardous reagents. The safe access to exotic reaction conditions such as high temperatures and pressures can be quite attractive as well, especially in the context of green chemistry.⁷⁵ Using flow reactors, green reagents which are normally less reactive than their classical, non-green counterparts can be used at temperatures above their boiling point, thus reducing reaction times.⁷⁶ Flow reactors also enable reactions in supercritical solvents such as supercritical methanol, ethanol, or acetonitrile,⁷⁷ or supercritical water⁷⁸ or CO₂.⁷⁹⁻⁸⁰ In particular, the latter is widely regarded as an attractive green alternative to conventional solvents due to its chemical and physical properties being tuneable across a wide range.⁸¹

Multistep syntheses in continuous flow are possible, in principle. Sophisticated total syntheses of natural products have been carried out partially⁸² or entirely⁸³ in continuous flow. To circumvent troublesome steps, batch processes have been integrated with continuous flow setups.⁸⁴ However, the difficulties regarding method development are compounded in multistep continuous flow syntheses. Reaction conditions of individual steps are no longer independent from each other as is the case in batch, where intermediates are commonly isolated before being used in the next step. Thus, all parameters of every reaction must be carefully considered not only with regards to the transformation at hand, but also to operations upstream and downstream of the step.^{60, 85}

Flow chemistry has been adopted in the manufacture of pharmaceutical products.⁸⁶ This is partially due to the abovementioned safety aspects, and partially due to the simple scalability. Beyond classical scale-up,⁸⁷ the relatively inexpensive nature of most flow reactors allows an alternative strategy known as numbering-up or scale-out⁸⁸ which involves running numerous small-scale reactors in parallel. Additionally, once a flow system has reached steady state there is no limitation to how long it is run, thus enabling a potentially indefinite continuous production of an active pharmaceutical ingredient or API.



Figure 10: Reconfigurable flow reactor for the production of four APIs. From Adamo, A.; *et al. Science* **2016**, 352 (6281), 61-7. Reprinted with permission from AAAS.

This realisation led researchers to investigate continuous end-to-end manufacturing solutions for APIs,⁸⁹ even including final dosage formulation.⁹⁰ However, those systems are usually tailored to the production of a single target compound and not universally useful. In the most impressive technology demonstration so far, a team at the Massachusetts Institute of Technology built a compact and reconfigurable flow system (Figure 10) capable of manufacturing four different APIs,⁹¹ however,

INTRODUCTION

extensive reconfiguration of the system was still required in order to switch between target molecules. Additionally, the points raised earlier regarding the finicky method development of multistep flow syntheses remain valid.

In the author's view, flow chemistry remains an enabling technology rather than an all-encompassing automation strategy. Aforementioned attempts at reconciling flow and batch or building reconfigurable flow systems are certainly promising, yet it remains questionable that they will eventually yield a truly universal automation strategy.

INTRODUCTION

4 AUTOMATED BATCH REACTORS

Relative to the aforementioned strategies, the automation of a pure batch synthesiser that is not parallelised was given very little attention. One of the first examples approaching an automated batch synthesiser was reported by Deming and Pardue in 1971⁹² where they attached a number of computer controlled syringe pumps and solenoid valves to a spectrophotometer cell which allowed them to collect mechanistic data in an automated fashion. This work inspired other groups to investigate automated batch synthesis. Charles Berkoff and Daniel Chodosh at Smith Kline & French reported a system capable of formulating, conducting and sampling a reaction in batch under full automation.⁹³⁻⁹⁶ Their publications are a fascinating insight into the technology at the time, since they describe in detail the electronics and the very limited computing power running their system. At the same time, Legrand *et al.* presented a similar system with similar capabilities (Figure 11).⁹⁷⁻⁹⁸

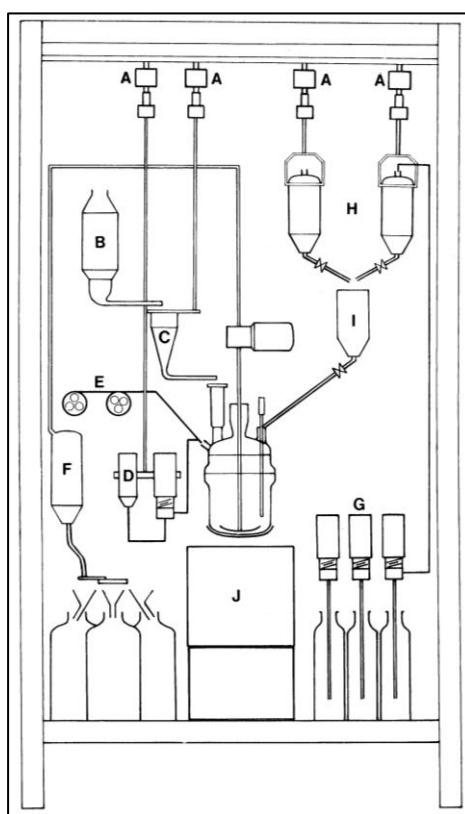


Figure 11: Schematic of Legrand and Bolla's apparatus.⁹⁸ A = strain gauges; B = powder dispenser; C = weighing vessel for powders; D = weighing vessel for liquid reagent with its transfer pump; E = peristaltic pumps; F = recuperation of the reaction mixture; G, H and I = solvent weighing and distribution; J = pneumatic jack for cooling.

INTRODUCTION

Those early systems were essentially automated reactors, intended to optimise reaction conditions and gather process data. Isolation of the product was not a major design feature, and neither team even considered conducting multistep reactions in their automated systems.

To the extent of the author's knowledge, the first automated multistep batch synthesis was carried out at the Conservatoire National des Arts et Métiers (CNAM) in France in 1987.⁹⁹ Their "Automated Versatile Modular Reactor" or AVMR (Figure 12) was equipped to add reagents, perform refluxing and distillation, as well as aqueous washing and liquid/liquid separation. Using this setup, the group performed a three-step synthesis, but did not isolate the crystalline end product automatically as their reactor lacked the capability for filtration. They correctly identified the need for versatility and the advantages of modular construction; however, it seems modularity in this context meant more a distribution of tasks rather than making individual components interchangeable. Indeed, in a 1989 paper the group presents an automated reactor for liquid/gas reactions¹⁰⁰ which appears to be an entirely new system bearing little resemblance to their original AVMR.

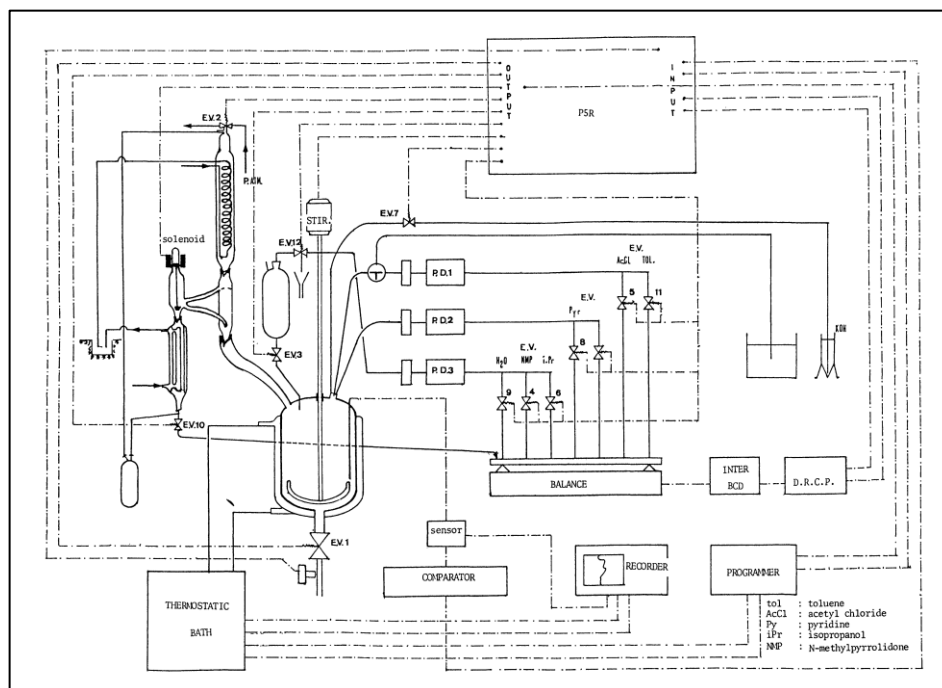


Figure 12: Schematic of the AVMR.⁹⁹

Meanwhile researchers at Takeda Chemical Industries in Japan were developing their own automated batch synthesisers. Their first generation system¹⁰¹ consisted of three reactors with reagent delivery system, a liquid/liquid separator, a preparative HPLC, and a freeze drying unit. They demonstrated the capabilities of their system by performing a reductive amination in a fully automated fashion. They later improved their setup in several iterations, arriving at a number of different platforms¹⁰² capable of performing most operations commonly encountered in organic synthesis. However, their system was bulky (between 180x180x40 cm and 180x200x70 cm) and employed complicated liquid handling systems largely driven by vacuum and solenoid valves, which required a staggering amount of tubing (Figure 13). Their control software was advanced for the time but tailored for every instance of their synthesiser, thus severely limiting the true modularity of the system.

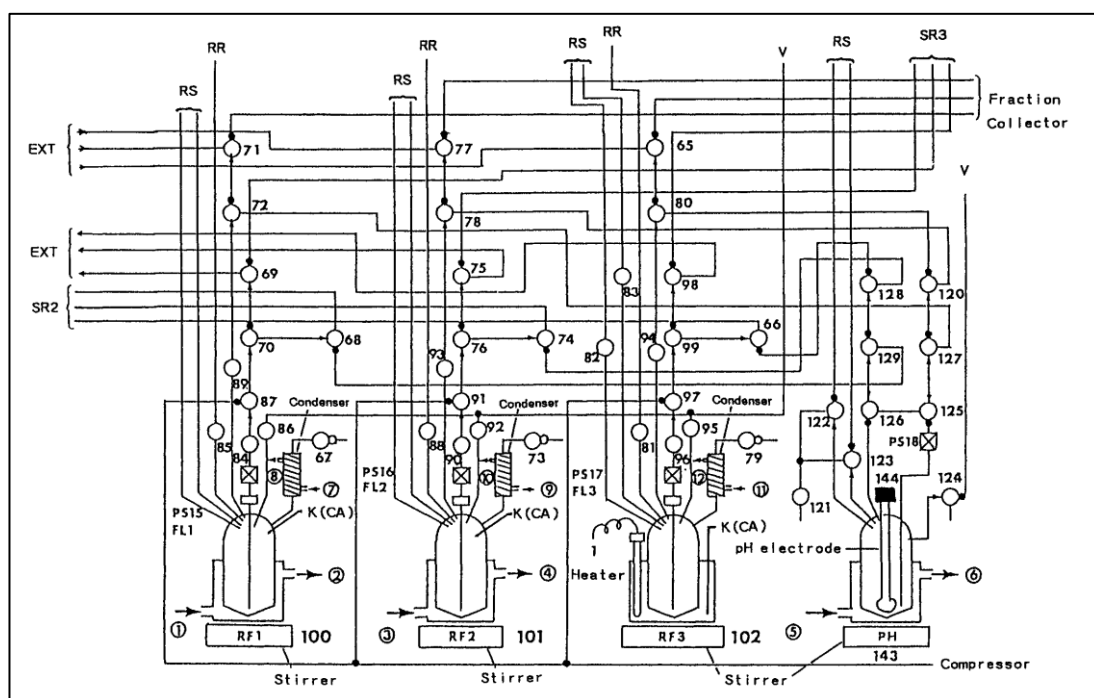


Figure 13: Schematic of the reaction unit of the Takeda platform,¹⁰² one of several subunits constituting their synthesis platform. Note the complicated tubing.

One of the researchers at Takeda, Tohru Sugawara, later joined the Takahashi research group. Together they developed a more advanced synthesis platform, the ChemKonzert (Figure 14).¹⁰³ Their system seems to build on many of the technologies developed at Takeda, including the liquid transfer via vacuum, and the complex

plumbing requirements. The most noteworthy addition is a centrifugal separator to speed up liquid/liquid separations. However, crucially, the system was not designed to perform evaporations, therefore precluding fully automated multistep sequences. Their control software *KonzertMeister* is not described in any detail, but it seems to be tailored to the particular setup of the synthesiser as well. Overall, the setup appears monolithic and not very flexible. The group has published numerous papers describing natural product syntheses assisted by the platform,¹⁰⁴⁻¹⁰⁸ but no significant advances on the instrumentation. It also has to be noted that, as mentioned above, the published syntheses relied on manual interventions in between the synthetic steps, thus reducing the utility somewhat. Nevertheless, it can be argued that the work of the Takahashi group neatly illustrates the usefulness of an automated lab-scale batch synthesiser in chemical research.

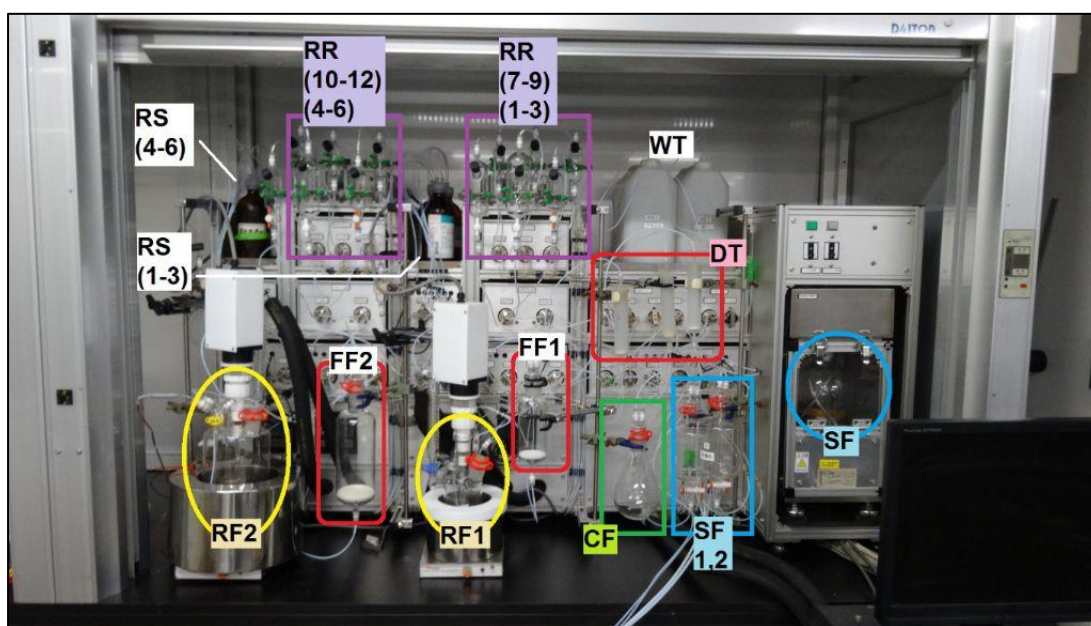


Figure 14: The Takahashi group's ChemKonzert synthesiser.¹⁰⁹ RF1 and RF2: reaction flasks; SF: centrifugal separator; FF1 and FF2: glass filters; RR1-RR12: reagent reservoirs; RS1-RS6: solvent reservoirs; DT: drying pads; CF: collection flask (for final product solution); WT: solvent tanks.

Examining the schematic of ChemKonzert (Figure 15), what is immediately obvious is that it lacks a dedicated evaporation module, which immediately precludes non-telescoped multistep syntheses. Furthermore, the architecture seems very

monolithic and purpose-built. It is hard to imagine how additional modules could be easily added without extensive changes to the overall system, as there seems to be no standardised expandable interfacing strategy. The authors claim a “unit concept” behind their design strategy, yet it remains unclear how those units could be customised independently, or how additional units could be added.

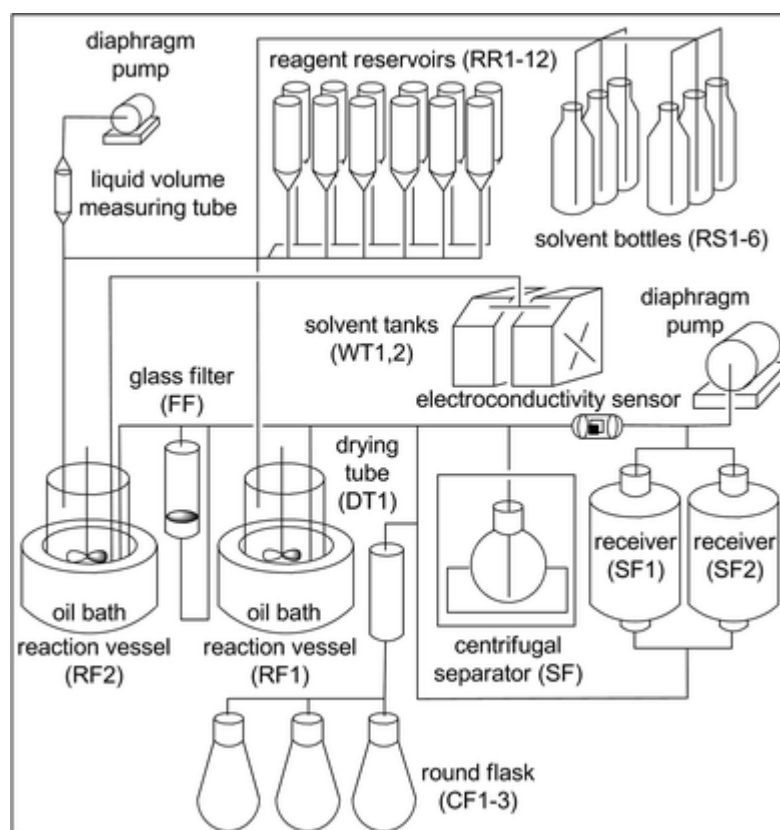


Figure 15: Schematic representation of the ChemKonzert platform. Reproduced from Ref. 106 with permission from The Royal Society of Chemistry.

To the extent of the author’s knowledge, Takahashi’s ChemKonzert is the only fully automated lab-scale batch synthesiser in use at the moment. The author’s best guess is that the drive for higher throughput and the relative ease of robotic automation shifted the attention of the mainstream automation community towards parallel synthesisers (see chapter 2), while the automation of larger-scale batch processes regressed to simply adding more instrumentation to a reactor, without bothering with work-up or isolation.¹¹⁰ At the time of writing, a number of automated reactor workstations are commercially available, such as the OptiMax by Mettler Toledo,¹¹¹ the Mya 4 by Radleys,¹¹² or the Atlas by Syrris,¹¹³ to name but a few. While those systems are undoubtedly useful for process research and development, the degree

INTRODUCTION

of automation they offer is relatively insignificant (addition of reagents, stirring, temperature programs, and the like), and unattended multistep syntheses are out of the question.

5 THE CHEMPUTER CONCEPT

Based on the assessment of the state of the art presented in the previous chapters, our group set out to create a new, universal, modular, lab-scale batch synthesis platform controlled by a flexible software suite. The working title for this platform was “The Chemputer”. This portmanteau of “chemical computer” was chosen as a reference to the proposed universality, similar to how the modern computer surpassed older, specialised calculating machines. To overcome the limitations of existing liquid handling solutions, the group started to develop new syringe pumps and six-way valves, and also implemented a novel architecture providing maximum modularity with minimal tubing. This “Backbone” architecture was conceived by a number of colleagues before I joined the project. Figure 16 shows a schematic representation of the Backbone. One pump and valve, respectively, represent a minimal unit. The Backbone can be elongated at both ends and conceivably closed to form a circle.

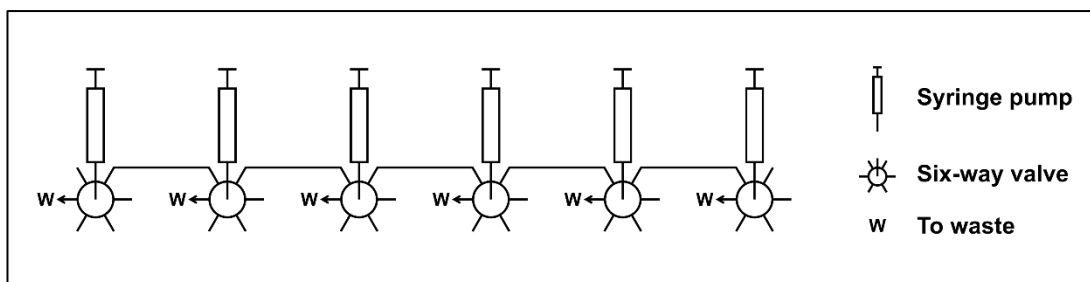


Figure 16: Schematic illustrating the Backbone architecture.

Figure 17 shows how to transfer liquid from a port on one Backbone unit to a port on another Backbone unit. The pump on the source unit aspirates the appropriate amount, then the valve on the source unit and the adjacent unit switch to the bridge, and the source pump and the adjacent pump move simultaneously to transfer the liquid contents from the source syringe to the next syringe. This process is repeated, and the liquid is moved along the Backbone until it reaches the destination unit, which in turn dispenses it to the destination port.

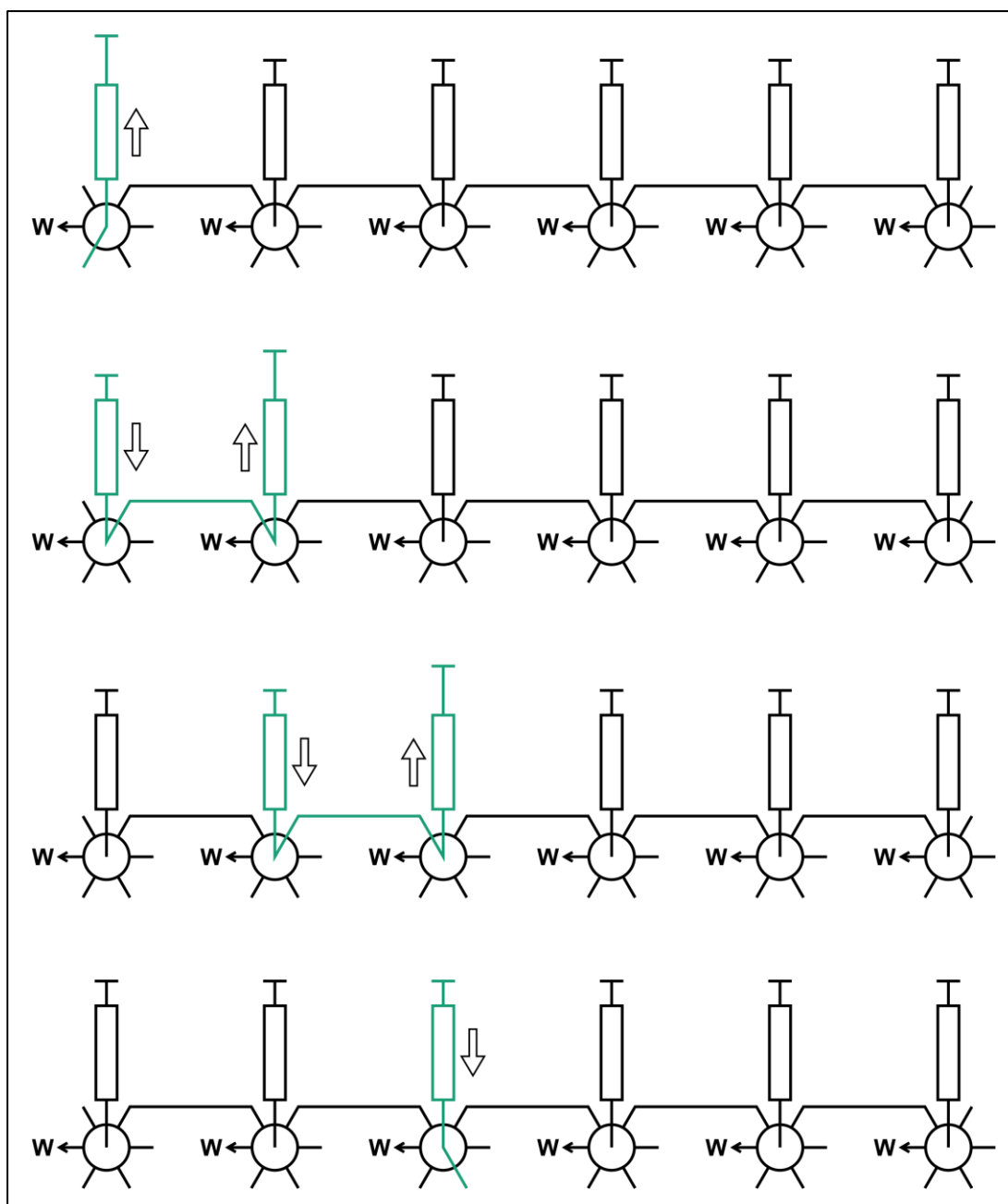


Figure 17: Moving liquids across the Backbone.

Initial efforts focused on relatively simple operations as the system's capabilities were severely limited. Multistep syntheses of small molecules were envisioned but at that point clearly beyond the capabilities of the system. The most complicated operation attempted was the solid phase synthesis of a peptide, which did work after significant troubleshooting, albeit at a yield that was much lower than that obtained in a commercial peptide synthesiser.

INTRODUCTION

The control software at that point was rudimentary. Drivers for the pumps, valves, and stirrer hotplates were implemented in Python, while the synthesis itself, including the knowledge about connectivity and reagent placement as well as all the transfer routines, was hard-coded in a single file containing close to 2,000 lines of Python code. A more flexible control suite dubbed “ChemOS” was being developed by Dr. Gerardo Aragon-Camarasa in parallel (see also chapter 7.3.2), but it wasn’t ready by the time I started work on the project.

AIMS

The first aim of the project was to develop the abstraction of a universal synthesis platform discussed in chapter 5 into a physical implementation. To that end, the aforementioned Backbone architecture should be combined with the required unit operation modules.

Thus, the available liquid handling hardware was to be improved to guarantee chemical resilience and general reliability. Also, the existing unit operation modules required further development, and new modules for further operations were to be developed.

The next aim was to improve the device firmware for the liquid handling hardware, and to further develop the overall control software. As mentioned in chapter 5, previous iterations of the platform used hard-coded scripts for control, so a more flexible, modular control suite was required.

With operational hardware and software in hand, the next aim was to thoroughly test the setup and verify the overall concept. Thus, the third aim was to perform three multistep syntheses covering an appreciable number of reaction classes on the platform and demonstrate the capabilities of the synthesis platform.

Beyond those immediate aims, a roadmap detailing the planned future work with this novel synthesis platform was to be drafted. In particular, the ongoing hardware and software development, more and more diverse chemistry, and the uptake in the group and the wider community should be assessed.

RESULTS AND DISCUSSION

6 HARDWARE DEVELOPMENT

During this work, a great deal of hardware development was undertaken in order to eventually yield a functioning synthesis platform (Figure 18). In particular, this development effort was focused on two broad fields: the liquid handling hardware constituting the Backbone, and the various unit operation modules. The initial state at the beginning of the project as well as the development and eventual operation of the individual components of the platform will be discussed in the following subchapters.



Figure 18: Photo of the synthesis platform developed.

6.1 PUMPS AND VALVES

6.1.1 PRIOR STATE OF THE ART

The in-house development of a new syringe pump was sparked by the desire to achieve higher flow rates than those offered by commercial pumps available at the time. Consequently, designing a six-way selection valve was a natural extension of that work, as a pump without a valve is arguably not very useful. Those efforts started long before I joined the group and were undertaken mostly by Dr. Stefan Glatzel, so I had no hand in many of the initial design choices.

RESULTS AND DISCUSSION

The principal design of the pump was very simple. A NEMA23 stepper motor with a lead screw moved a carriage along a linear guide rail. The barrel of a glass syringe was mounted to the motor and guide rail, while the plunger was mounted to the moving carriage. The home position was established by a Hall effect sensor in the base and a magnet in the carriage. The earliest prototypes featured a horizontal syringe (Figure 19) similar to many commercially available models, while later versions had an upright design with the outlet at the bottom (Figure 20). Control was achieved *via* an Arduino and an external stepper motor driver. Early versions can be viewed as simplistic imitations of the archetypical, commercially available syringe pump design.

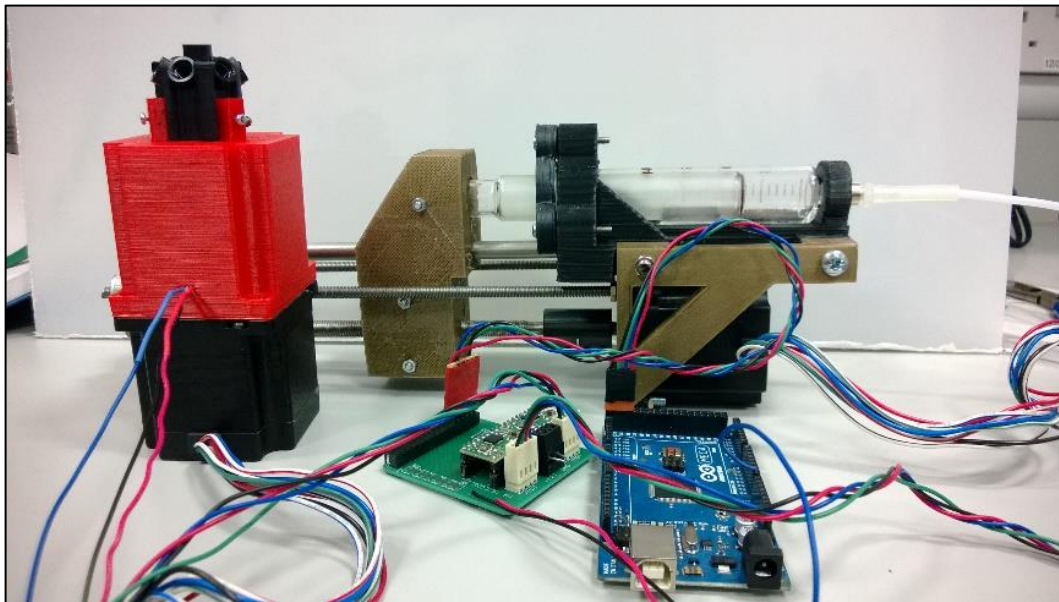


Figure 19: Early prototypes of the six-way valve and the syringe pump, together with an Arduino and a motor driver PCB to control the devices.

All connectors were manufactured from polylactic acid (PLA) on a Fused Deposition Method (FDM) 3D printer. While 3D printing in principle has many advantages, in this case it negatively affected the mechanical design in fundamental ways. First, the poor mechanical properties of 3D printed PLA parts required a bulkier design. Then, the poor resolution of the printer required many key dimensions to be specified with wide tolerances, leading to severe problems with misalignments in the assembly. Additionally, the relative freedom in terms of design that 3D printing offers meant

that manufacturability in a more conventional context was never a consideration when many early design choices were made.

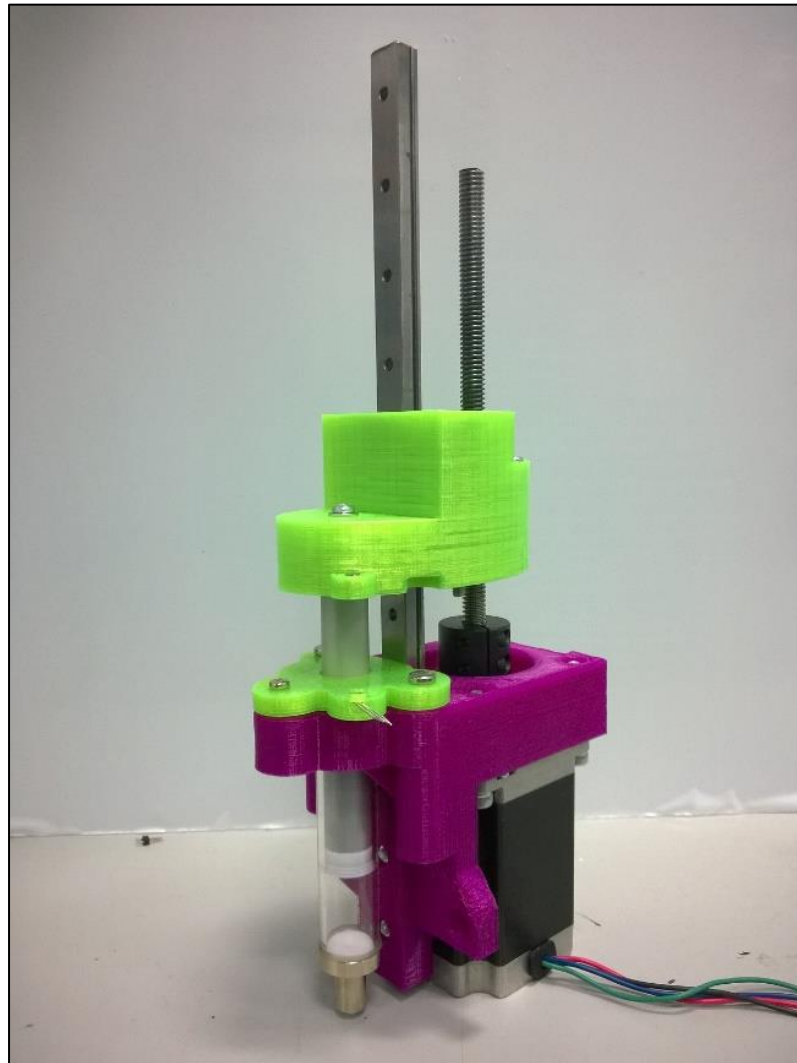


Figure 20: One of the first prototypes of the current pump design.

The motor connector was later modified to accommodate a range of syringes, using inserts and cover plates of different sizes. The purchase of a Stratasys Connex 500 polyjet 3D printer allowed manufacture of the parts from a proprietary photopolymer with mechanical properties similar to unplasticised PVC, thus enhancing the overall mechanical properties of the device. Figure 21 shows the components of one syringe pump. This image largely reflects the starting point of my involvement in the project.

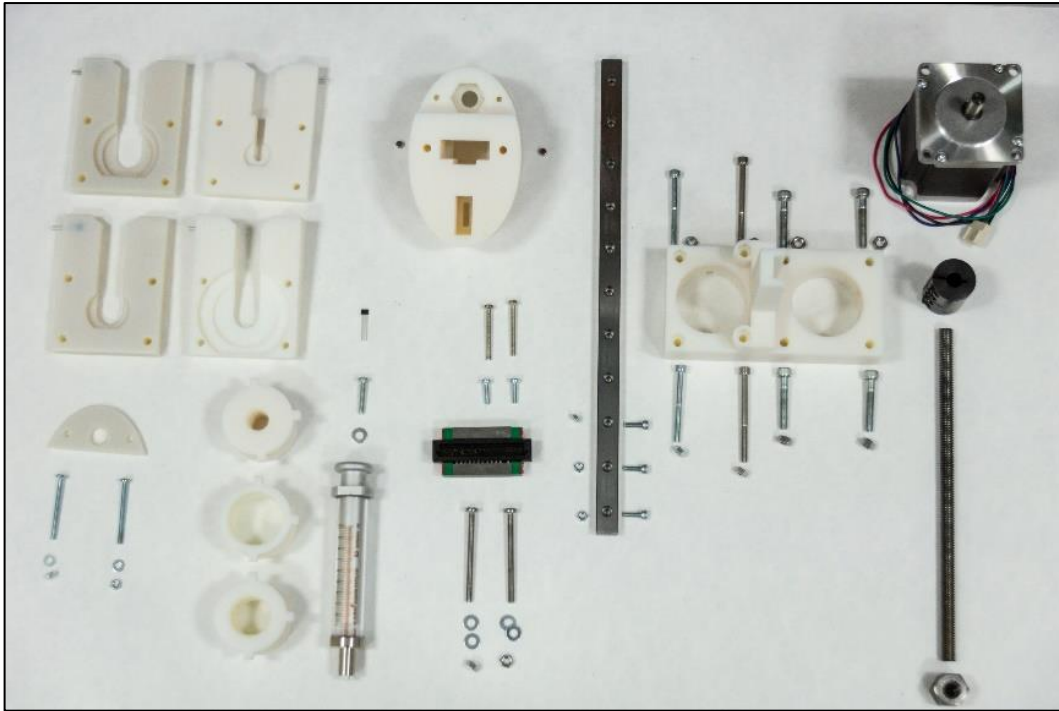


Figure 21: Components of the syringe pump, late 2014.

The valve design is based on the commercially available V-240 six-way selection valve manufactured by IDEX Health & Science. This valve features a central inlet and six selectable outlet ports fitted with $\frac{1}{4}$ -28 UNF threads. Originally the valve comes with a hand wheel which is attached to the shaft with a grub screw. This hand wheel was replaced with a 3D printed connector that allowed mounting the valve onto a NEMA23 stepper motor. A 3D printed housing held the valve body in place relative to the motor (see Figure 22). Positioning was achieved by six magnets embedded in the motor connector, and a Hall effect sensor integrated into the housing.

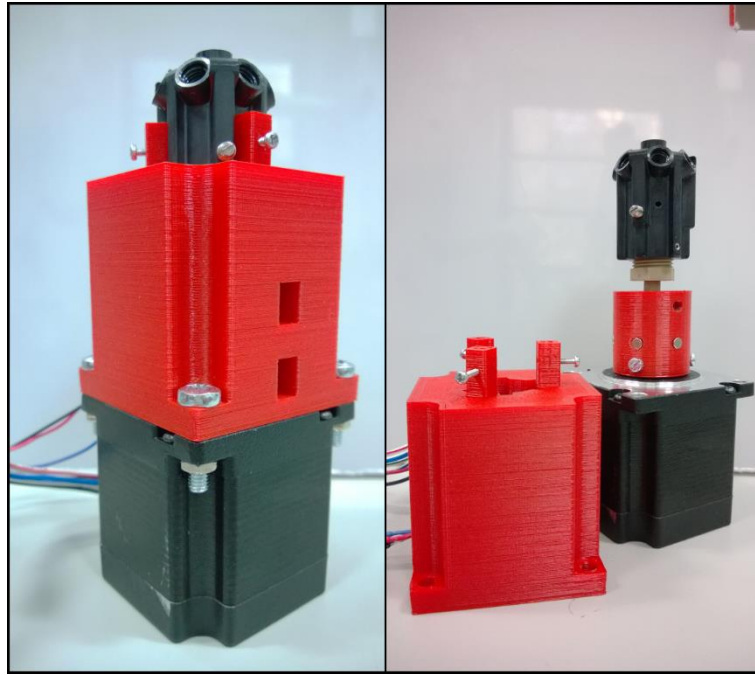


Figure 22: Early prototype of the six-way valve. Left: full assembly. Right: partially disassembled, showing the motor connector and the positioning magnets.

Only a few alterations were made to this design before I joined the project. The size of the magnets was decreased to improve positioning accuracy. The overall shape of the motor connector was initially optimised for minimal material use, but later simplified to a plain cylindrical shape. The most noteworthy change was made to the way the Hall effect sensor was held in place. Initially, a through hole technology (THT) type sensor was used (see Figure 28). The sensor body was press-fitted into the valve housing, the pins were extended through holes to the outside, and connected to a cable either by direct soldering, or some sort of female connector. However, mechanical strain on the cable would regularly damage the sensor, and exchanging it proved difficult when it was simply embedded into the housing. Thus, a holder for the sensor was designed in such a way that it would slip into a corresponding slot in the housing from below and be held in place by the motor. In case of a sensor failure, the whole piece could be discarded and replaced. This system, and the wider issue of the Hall effect sensor, will be discussed at length in chapter 6.1.4.

Figure 22 and Figure 23 show, except for the abovementioned changes, the state of the six-way valve when I joined the project.

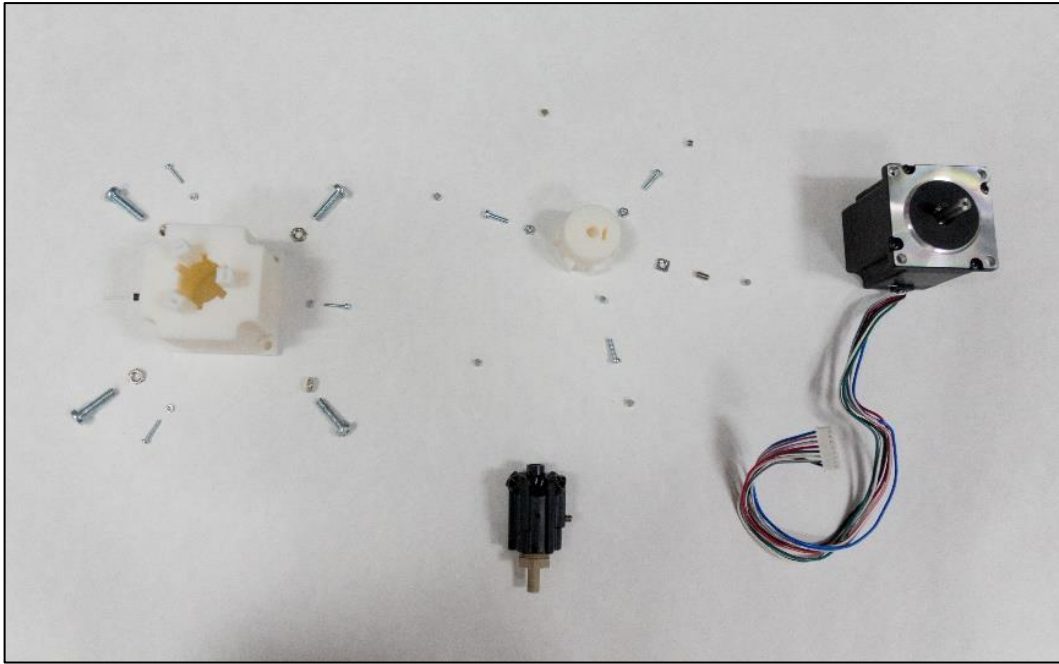


Figure 23: Components of the six-way valve, late 2014.

The Arduino controlling the devices was later replaced with a custom PCB using Power over Ethernet (PoE) to both power and control the boards. This interesting design choice meant that only one cable was required per device, and multiple devices could be connected to a commercially available PoE switch, thus making the overall experimental setup neater and tidier. The PCB went through multiple design iterations both in-house and externally, the currently used model was designed by a contractor.

The basic design of the PCB follows the successful Arduino design.¹¹⁴ It consists of two boards, a mother board and a motor driver shield. The mother board features an Atmel ATxmega128A4U microcontroller, a WIZnet W5500 ethernet chip, an RJ45 socket for the ethernet connection, and periphery for power control and distribution. It also features two rows of female pin headers which allow the connection of various daughter boards or shields. The motor driver shield features a Trinamic TMC262 stepper motor driver and the associated MOSFET stage.

6.1.2 TRANSFER TO AUTODESK INVENTOR AND INITIAL IMPROVEMENT EFFORTS

The initial development work described above was largely done in Autodesk AutoCAD, which is a CAD application initially developed for 2D applications; and OpenSCAD, which is an open source 3D CAD application which uses a scripting language to build solid bodies. Neither application is suitable for a larger engineering project as the 3D modelling capabilities of both programs are limited, and neither offers an assembly feature. Thus, the parts were ported to a suitable engineering CAD package. After some consideration Autodesk Inventor was chosen, which offers parametric and freeform modelling, assembling components and inspecting fit and function digitally, as well as the creation of construction drawings, among many other features. It directly competes with SolidWorks or Solid Edge, but in this case previous experience with Inventor made it the best solution.

Instead of simply importing the existing STL files as solid bodies, it was decided to draft every part from scratch, as this would allow easy alterations later on. Also, a preliminary assembly of just the printed components was created to verify fit and alignment. This flagged up several problems with the existing models, as some key features on the pump carriage and the pump motor connector did not line up properly. After fixing those problems, a version number was embossed onto every part and a set of proofs was printed. The version number turned out to be an invaluable feature, as previous design iterations left a plethora of old parts in their wake, which led to some confusion. It was also found that many clearances were wider than required, presumably because of the inferior performance of the FDM printer used before, which further complicated proper alignment of all parts during assembly. Thus, most clearances were tightened to around 50-100 μm and experimentally verified a snug fit by printing proofs.

The pump design initially used a number of different screw lengths without any justification (see Figure 21). This in turn meant a larger number of screws had to be sourced and stocked, complicating the life cycle management of the pump. The block of the linear guide rail was mounted to the pump carriage using two M3x10 and two M3x30 for no apparent reason (see Figure 24). It was decided to deepen the lower holes in the pump carriage to allow the use of four M3x10 instead (Figure 27), as this

screw size was already used for mounting the guide rail to the motor connector. The cover plate holding the syringe in place was mounted by two M4x35 and two M4x50. Again, the geometry was adapted to allow the use of four M4x35 instead, further reducing the overall number of different components.

6.1.3 IMPROVING THE ALIGNMENT

As mentioned before, the early models of both the pump and the valve suffered from a range of problems due to improper alignment of the components. If the valve housing (Figure 25 C) was not carefully aligned with the motor prior to assembly, a process which was done largely by eye, the rotating connector would scrape against the housing. The motor connector itself had to be carefully positioned at a precise height using a calliper or feeler gauge so the magnets were positioned at the same height as the Hall effect sensor, otherwise the positioning accuracy would drop significantly. Tightening the screws holding the motor connector to the shaft had to be done carefully to ensure the motor connector wasn't tilted, which would lead to it scraping against the housing and potentially getting stuck. Nevertheless, the motor connector would still often precess, which made the valve head move around visibly. To make matters worse, the vibrations caused by heavy use, exacerbated by mechanical forces caused by this precession or scraping, would regularly loosen the screws holding the assembly together. This would lead to increased movement within the assembly, worsening the condition, sometimes up to a point where the valve would cease to function entirely.

The pump was suffering from similar problems. The motor had to be centred by eye as well. Additionally, earlier models used a stepper motor with a plain shaft, and a mechanical coupler to attach the lead screw. Those connections were rarely coaxial, leading to eccentricity and precession of the lead screw. The plunger of the syringe was attached to the carriage with a screw in a slot, allowing adjustments along one axis (Figure 24). This was found to be necessary in early design iterations, as the FDM printer simply lacked precision. Unfortunately, this meant that the plunger had to be carefully aligned by eye and feel, and improper alignment would often cause the plunger to scrape against the syringe barrel and get damaged. Inevitable misalignments would again lead to unwanted mechanical forces and excessive

vibration, working screws loose to the point where they fell out of the device after several hundreds of moves.

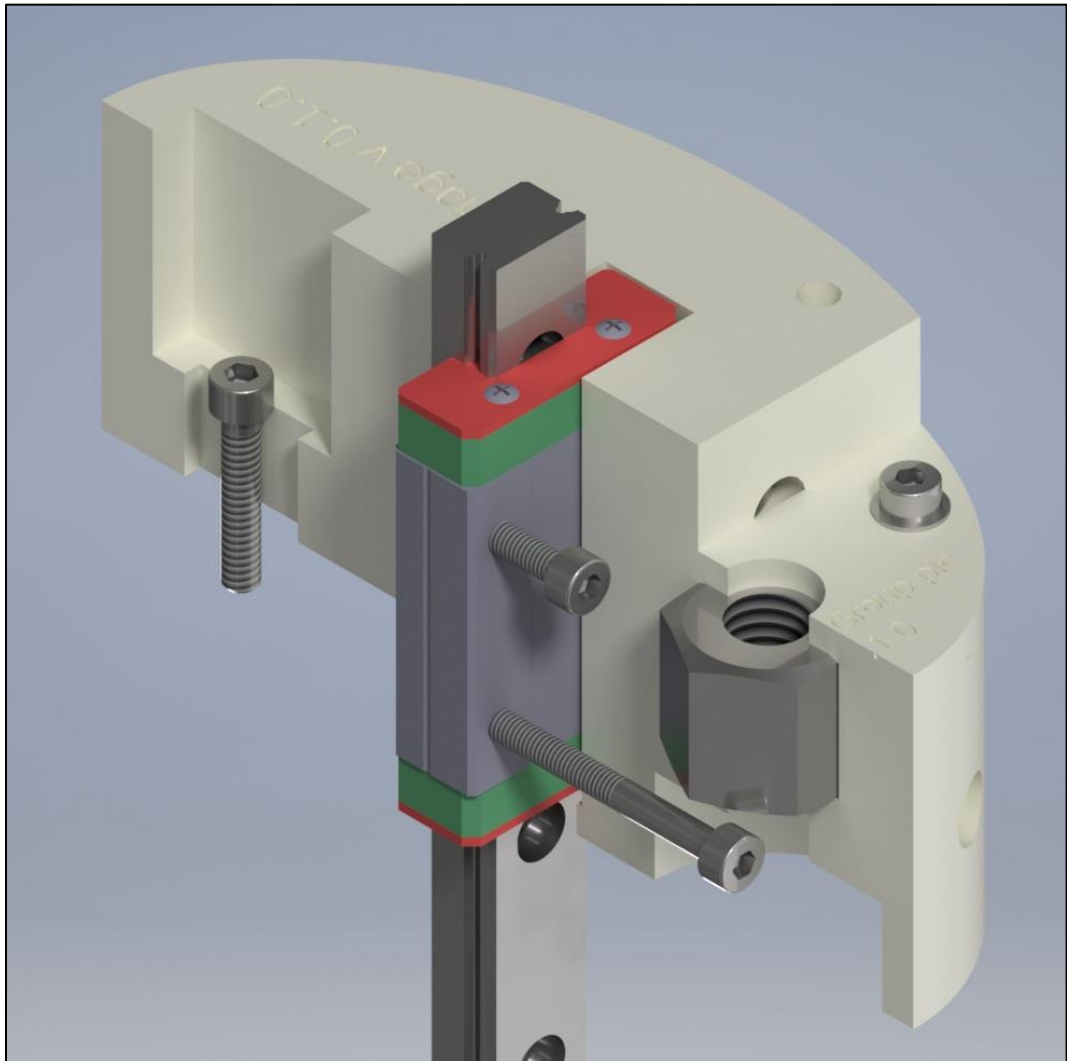


Figure 24: Section view of the old pump carriage. Note the slot for the screw holding the syringe plunger (top left), the dissimilar screw lengths for the linear guide block, and the hexagonal lead screw nut which is just clamped in place by a thin plate.

For the valve, three critical alignments were identified. First, the internal cylinder of the housing must be coaxial with the motor shaft. Second, the motor connector must be coaxial with the motor shaft. Third, the distance between motor connector and motor must be correct. To address the first alignment, the base of the housing was widened, and a raised skirt snugly fitting the faceplate of the motor was added (Figure 25 D). Thus, the housing could no more be translated or rotated in relation to

the motor. In the course of this design iteration, nut traps were added to the housing to simplify assembly. Previously, the screws would be inserted from above and the nuts placed on the motor side (Figure 25 A). This way, the nuts had to be secured against rotation when tightening the screws. The confined space made the nuts awkward to hold with a spanner, so mostly a large flathead screw driver was wedged between the nut and motor, which was a fiddly and awkward operation. In the new design, the nuts would be inserted into hexagonal recesses in the housing (Figure 25 B), and the screws would be inserted from below and could be comfortably tightened.

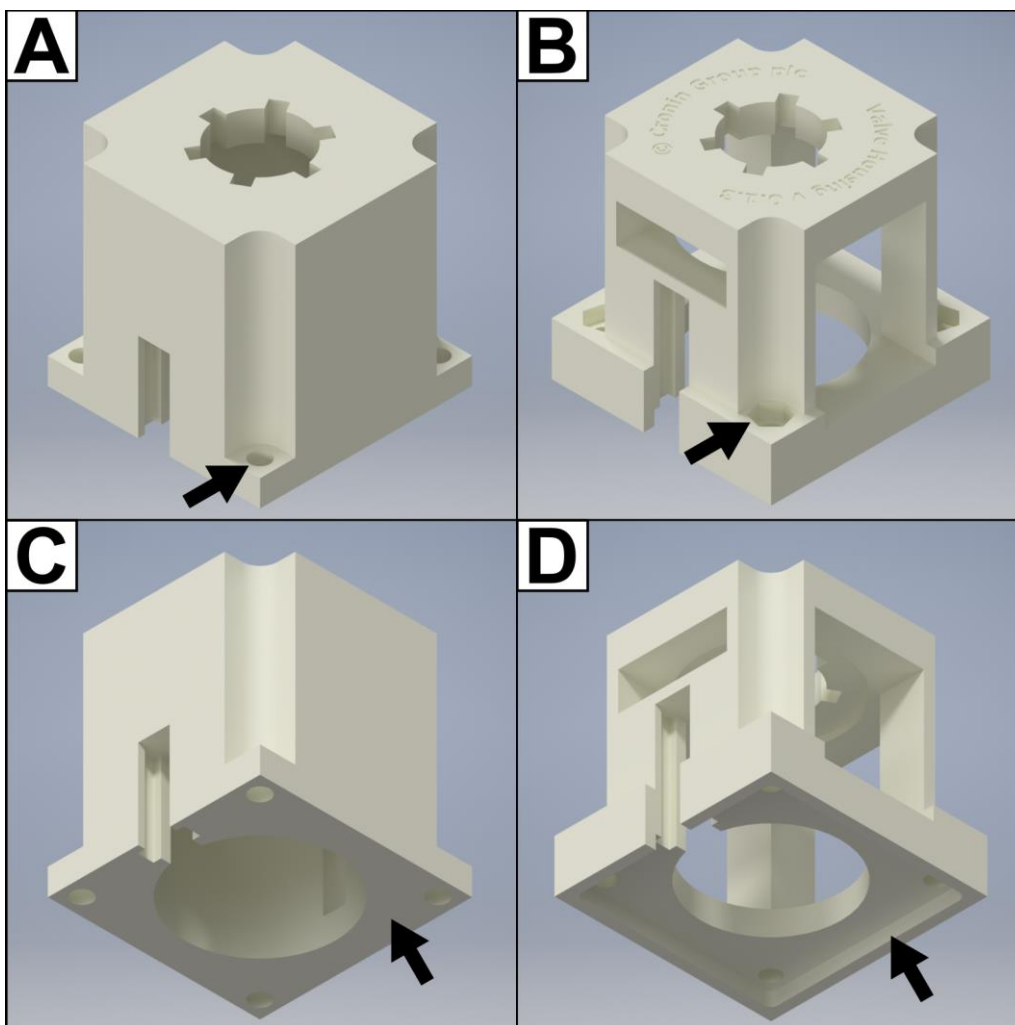


Figure 25: Valve housing. A) initial design, showing the plain mounting holes (arrow). B) improved design, showing the hexagonal nut traps (arrow) as well as the embossed version. C) initial design, showing the plain underside (arrow). D) improved design showing the raised skirt (arrow) for locating the motor. The large openings in the new design allow easy access to the motor connector for monitoring and trouble shooting.

The motor connector alignment was improved by decreasing the clearances, and by repositioning the D cut. Initially, the central bore of the motor connector had a nominal diameter of 6.5 mm. The NEMA standard specifies a nominal outer diameter of .250" or 6.35 mm for the shaft of a NEMA23 motor, which leaves a 75 μm gap all around, which is enough to visibly tilt the connector. Practical experience with the Connex 500 3D printer showed that tolerances as small as $\pm 10 \mu\text{m}$ were achievable, so the nominal diameter of the bore was decreased to 6.4 mm. This provided a very snug fit of the motor connector and eliminated any angular play.

The motor used for the valve features a D cut to efficiently transfer the torque to the valve *via* a matching feature in the motor connector. This D cut could also serve as datum point for the vertical alignment of the motor connector by repositioning the ridge of the feature inside the connector to rest on the feature in the motor shaft. During assembly, the connector now just had to be pushed all the way down and secured by tightening the screws.

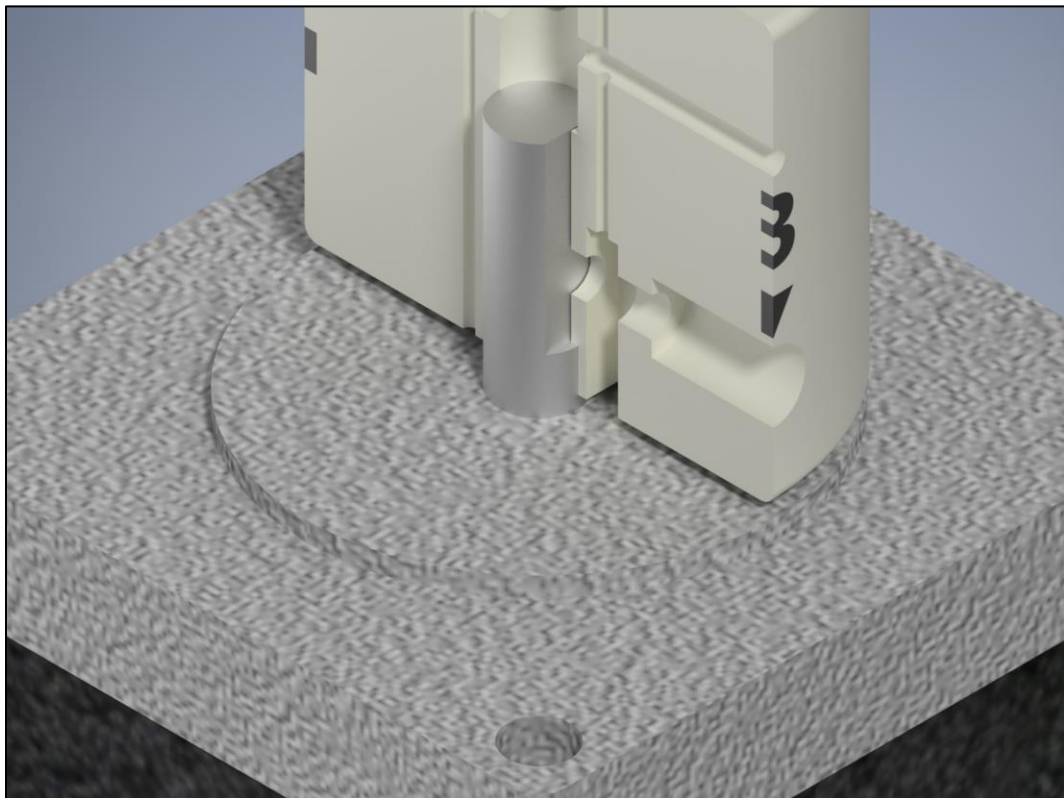


Figure 26: Section view of the new motor connector. Note how the printed part sits on the D cut in the motor shaft.

RESULTS AND DISCUSSION

Those alterations greatly simplified the assembly of the valves, and at the same time improved the reliability. Strong vibrations were still a concern, though, particularly because the screws in the motor connector regularly worked themselves loose and subsequently were propelled outwards by centrifugal forces during valve movements, leading to jams. To address this particular issue, thread locker was specified for all screws inside the motor connector, while the screws connecting the housing to the motor were fitted with locking washers. This strategy successfully mitigated the detrimental effect of vibrations and led to highly reliable operation for thousands of moves.

For the pump, the central axes of the motor and the syringe were identified as critical alignments. To properly line up the centre points for both components on both the carriage and the motor connector, the linear guide rail and block provided two convenient datum surfaces. The motor connector was fitted with a skirt as described above for the valve housing, precisely locating the motor. The motor itself was replaced with a model featuring an integrated lead screw, thus eliminating the linear coupler. The hexagonal lead screw nut used previously was replaced with a flanged lead screw nut featuring four threaded M3 holes. This in turn allowed simplification of the carriage.

The position of the syringe barrel relative to the motor connector is determined by the cover plate. The end cap of the syringe is wider than the glass barrel, so the sleeve insert has to provide sufficient clearance to allow the end cap to pass through. The cover plate however could be accurately moulded to fit the flange of the syringe, providing accurate positioning. In order to accurately and repeatably position the cover plate relative to the motor connector, locator pegs were added to the connector, while corresponding recesses were added to the cover plate. Shape and location of those pegs went through some iterations, but the overall concept was found to work well. On the carriage, the slot for the screw holding the syringe barrel in place was replaced with a tight clearance hole (Figure 27).

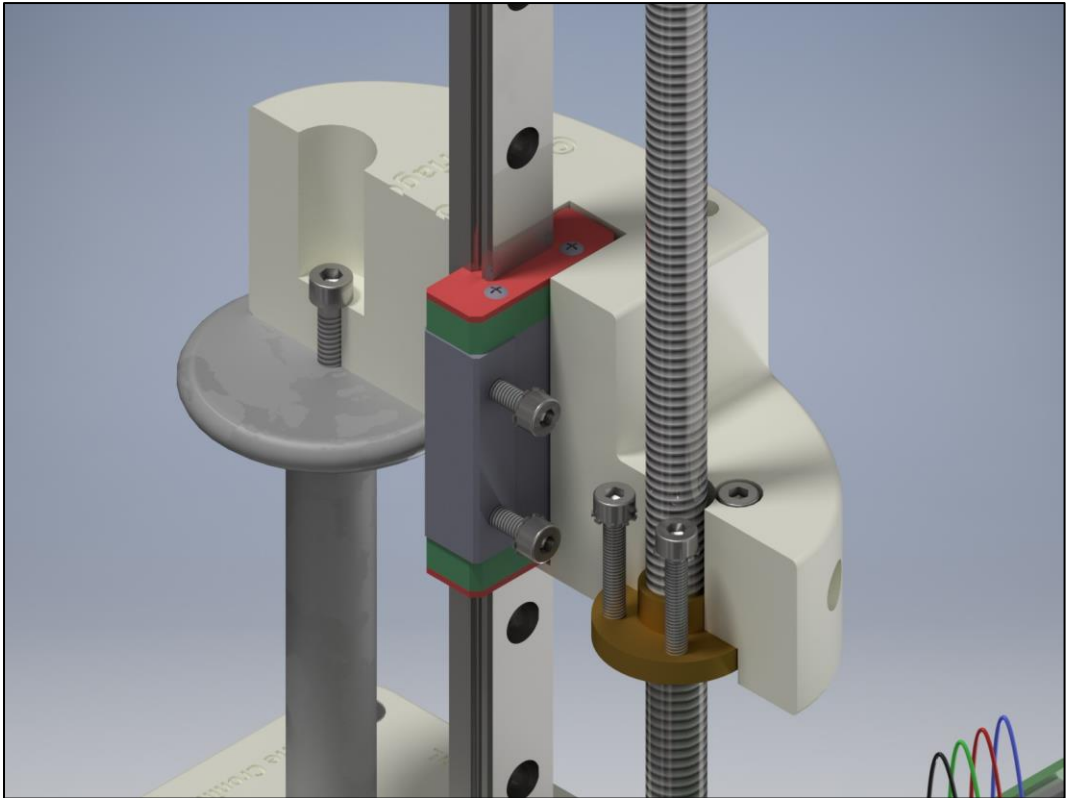


Figure 27: Section view of the new pump carriage. Note the new lead screw nut, the clearance hole for the syringe bolt, and the locking washers on most screws.

To address the vibration issue, all permanent screws were fitted with locking washers. Only the four screws securing the cover plates and the screw holding the syringe barrel were left without locking washers as they have to be undone when the syringe should be swapped. With those safeguards in place, stress tests with several thousand random moves did not dislocate any screws.

6.1.4 THE EVOLUTION OF THE HALL EFFECT SENSOR

Both pump and valve used analog bipolar Hall effect sensors for positioning. The first versions of the devices used a three pin through hole package sensor (Figure 28). The cover plate of the pump and the valve housing were fitted with a small recess to hold the sensor and three small holes to keep the pins separate. The earliest designs had only a slot, so short circuits of the sensor were a common occurrence.

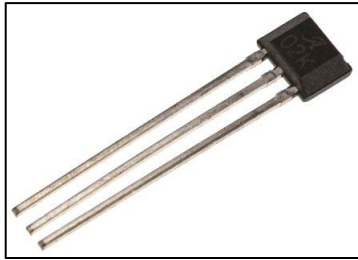


Figure 28: Allegro Microsystems A1302 analog bipolar hall effect sensor, through hole package.

This mode of mounting the sensor had numerous disadvantages. First, if the recess was too large, the sensor could move, and the reading would shift. This was sometimes counteracted by gluing the sensor in place (Figure 29), which produced problems when the sensor malfunctioned. If, however, the recess was too tight, excessive force was required to push in the sensor, often damaging the fragile component. Furthermore, there was no convenient way of interfacing the pins with a cable leading to the control PCB. The most reliable connection was directly soldering the pins to the wires; however, the sensors were known to break or malfunction frequently, and replacing them would entail desoldering and resoldering the wires every time. Alternatively, various female connectors were used, however, the small diameter of the sensor pins made most of those connections unreliable. Also, neither the pump nor the valve design at the time provided any means to firmly attach any connector to the device, so any mechanical strain on the cable would either unplug it or tear out the pin. As the pins simply protruded from the side of the devices in a rather exposed location, the cable got commonly snagged or squashed during handling, damaging the sensor.

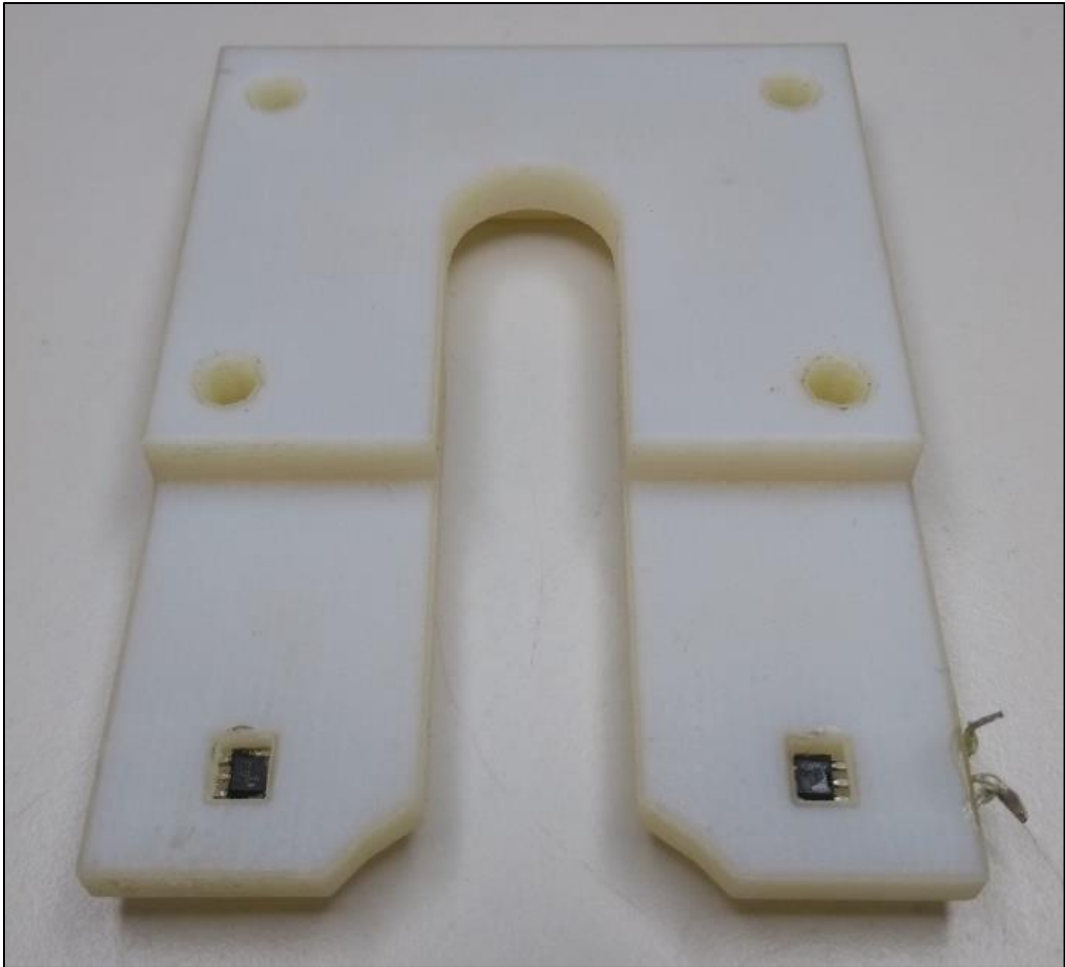


Figure 29: Old cover plate with the Hall effect sensor glued in place. The disadvantages of this mode of mounting the sensor are readily apparent from the poor condition on the sensors: the pins of the right sensor are bent, while the pins on the left sensor are sheared off. This sort of malfunction was commonly encountered even after little use.

The least reliable and inconvenient of the abovementioned ways of contacting the sensor was the use of a three-pin female crimp connector which was then glued to the side of the device. Although the pins were too thin for the connector, when fully inserted the mechanical forces would bend them against the sides of the female connector and ensure a somewhat durable electrical connection. By gluing the connector to the device, mechanical strain on the cable would not be translated to the pins. Obviously, this was a rather permanent solution, and in case of a sensor malfunction the entire part had to be discarded.

RESULTS AND DISCUSSION

Clearly, a better solution was needed. The obvious way of fixing many of the problems was to continue using the female crimped connectors, but to encase the entire assembly consisting of connector and sensor within the device. To that end, the cover plate of the pump and the holder insert of the valve were redesigned.

The holder insert for the valve was split along the centre, and a cavity was added to accommodate the sensor and connector (Figure 30). The wires were routed out through three circular openings on the side which firmly held the cable in place and thus absorbed any mechanical strain. When assembling the valve, the sensor was inserted into the connector of the cable, both were then inserted into the holder, and the whole holder assembly was slid into the housing. It was found that this mounting mode held the sensor in place reliably and protected it against mechanical damage. In case of a malfunction, the sensor and/or the cable could be replaced easily.

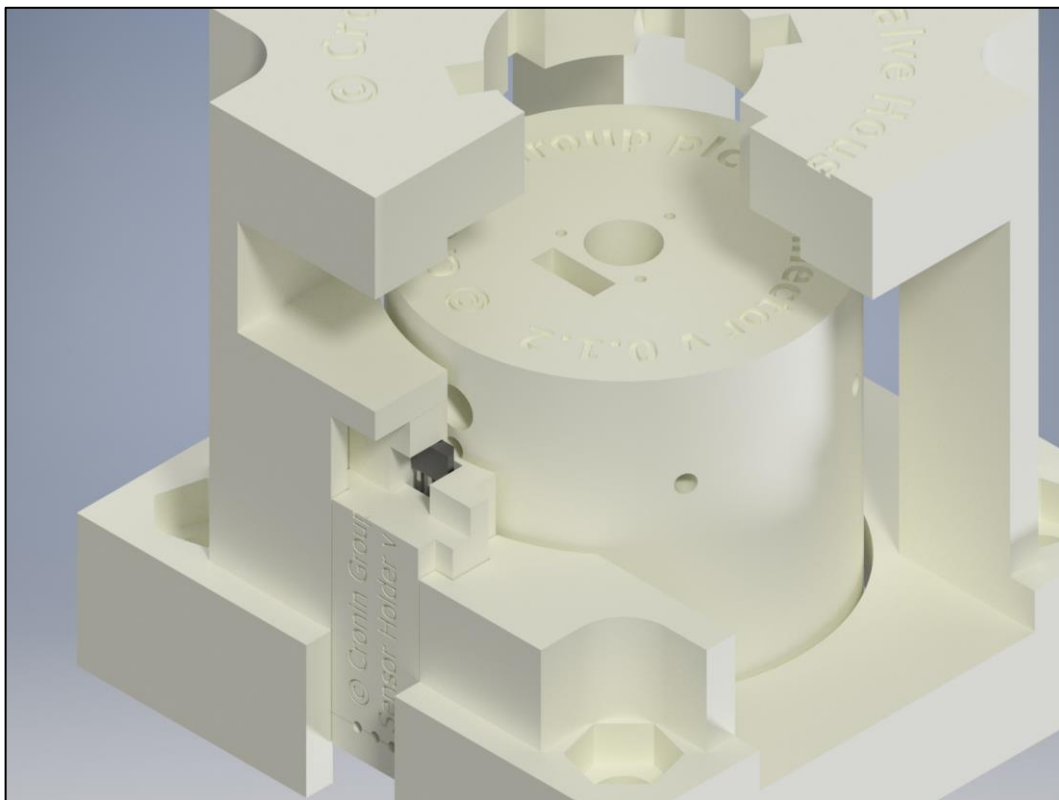


Figure 30: Section view of the valve showing the Hall effect sensor (black) embedded inside the two halves of the holder.

The pump cover plate was elongated towards the back and a shaped recess was added to accommodate the sensor and connector. A shaped wedge was then inserted from below to hold the assembly in place, and the plate was mounted onto the pump (Figure 31). While developing this solution, another design flaw was discovered. The home position of the pump is determined by the distance between the top of the syringe barrel and the top of the plunger. Unfortunately, when ordering the custom-made syringes, no attention was given to unifying this distance across different syringe sizes. Consequently, the home position was different for every syringe size. This is the reason why the Hall effect sensor had to be mounted into the cover plate rather than directly onto the motor connector, as the different thicknesses of the plate would position the sensor at the appropriate height. An inquiry with the syringe manufacturer revealed that unifying the plunger lengths would be prohibitively expensive, so instead of keeping the position of the barrel constant and varying the home position of the carriage, it was decided to keep the home position constant and vary the height of the barrel *via* the different sleeve inserts.

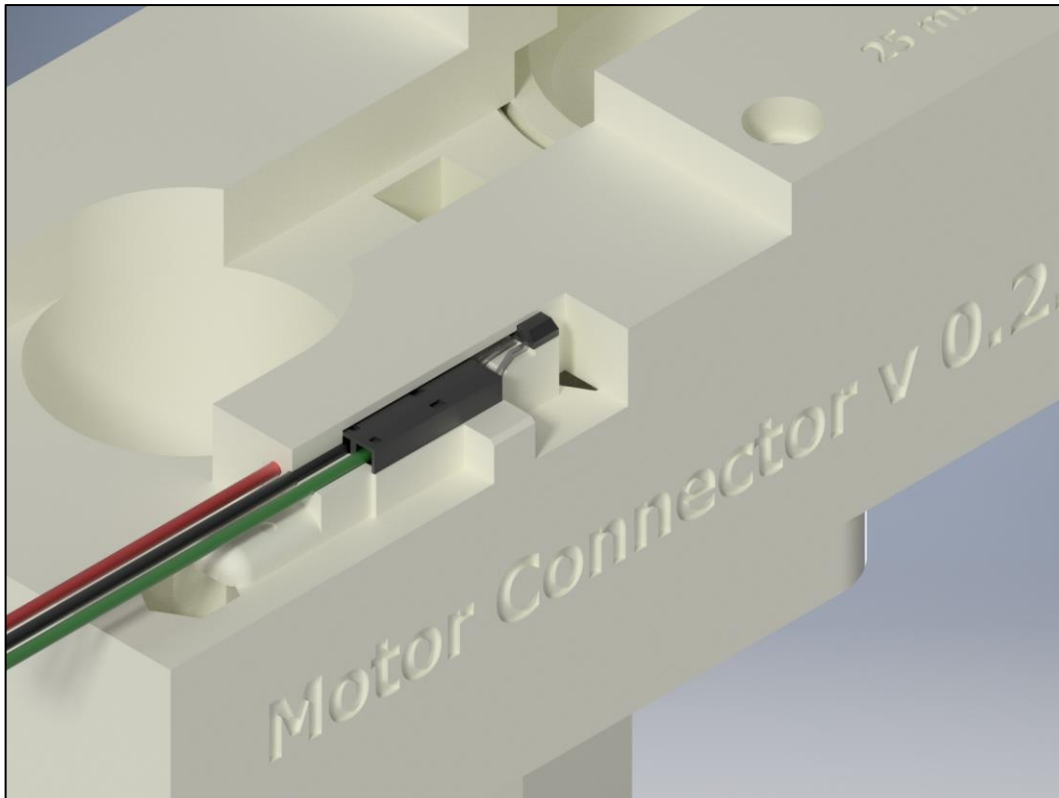


Figure 31: Section view of the pump showing the Hall sensor and crimp connector encased in the cover plate.

The design of the new cover plate went through a few iterations because early designs were prone to crack when the carriage crashed against the base, which happened occasionally when the sensor malfunctioned. By carefully sculpting the sensor cavity, reinforcing the plate surrounding it, and reshaping the motor connector to support the plate all around, this cracking problem was eventually solved.

This new solution performed significantly better than the previous, but sensor-related malfunctions were still a common occurrence, and particularly in the case of the valves it often took a while to pinpoint the failure and initiate repair. Also, integrating the sensor into the cover plate of the pump meant that every time the syringe was swapped, the sensor had to be repositioned, which was a fiddly process. Still not satisfied, another, completely different solution was devised.

The particular Hall effect sensor used was also available in a surface mounted device (SMD) form factor. Therefore, it was decided to design a small PCB featuring the Hall effect sensor, three 2.54 mm pitch pin headers which were designed to fit the female connector previously used, and two M2 mounting holes. This PCB could then be mounted onto the pump using the mounting holes (Figure 32) or slid into the valve housing similar to the holder used previously (Figure 33). The pin headers provided reliable electrical connection, while the sensor was positioned properly without suffering mechanical strains. Since the position of the sensor relative to the pump body was now harmonised due to the different sleeve inserts, the sensor could now be mounted directly onto the motor connector. This allowed the cover plate to be shrunk significantly and made swapping the syringe a more straightforward process.

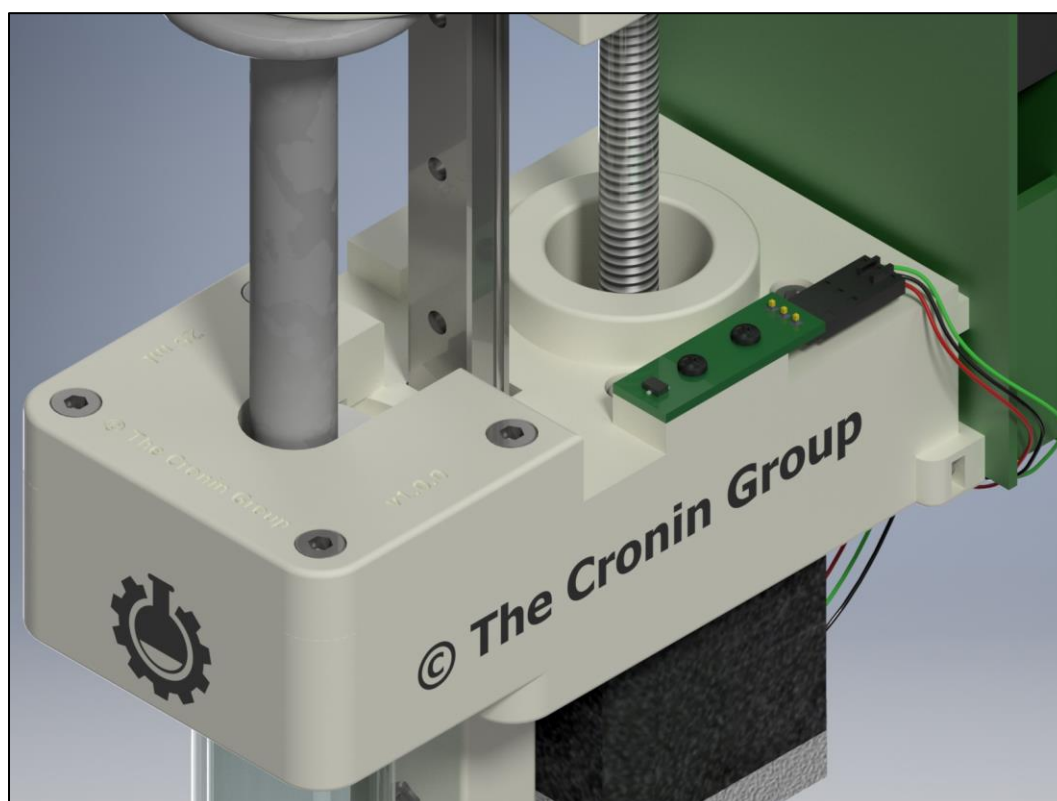


Figure 32: Hall effect sensor breakout board mounted onto a pump.

This solution performed exceptionally well and devices featuring this Hall sensor breakout board were used in all successful syntheses (see chapter 8). It was found, however, that crimping the connectors, while in principle a straightforward process,

proved challenging to some. Multiple batches of cables were thus ordered from a contract manufacturer, but the cost was significant. Suitable ready-made cable assemblies containing appropriate connectors could unfortunately not be sourced. Thus, it was decided to instead buy cable assemblies consisting of two Molex MicroClasp connectors, cut the cables in half, and directly solder the leads to the Hall sensor breakout boards. Previous experience showed that with the breakout boards, the sensor itself was relatively reliable, so a permanent connection to the cable was warranted.

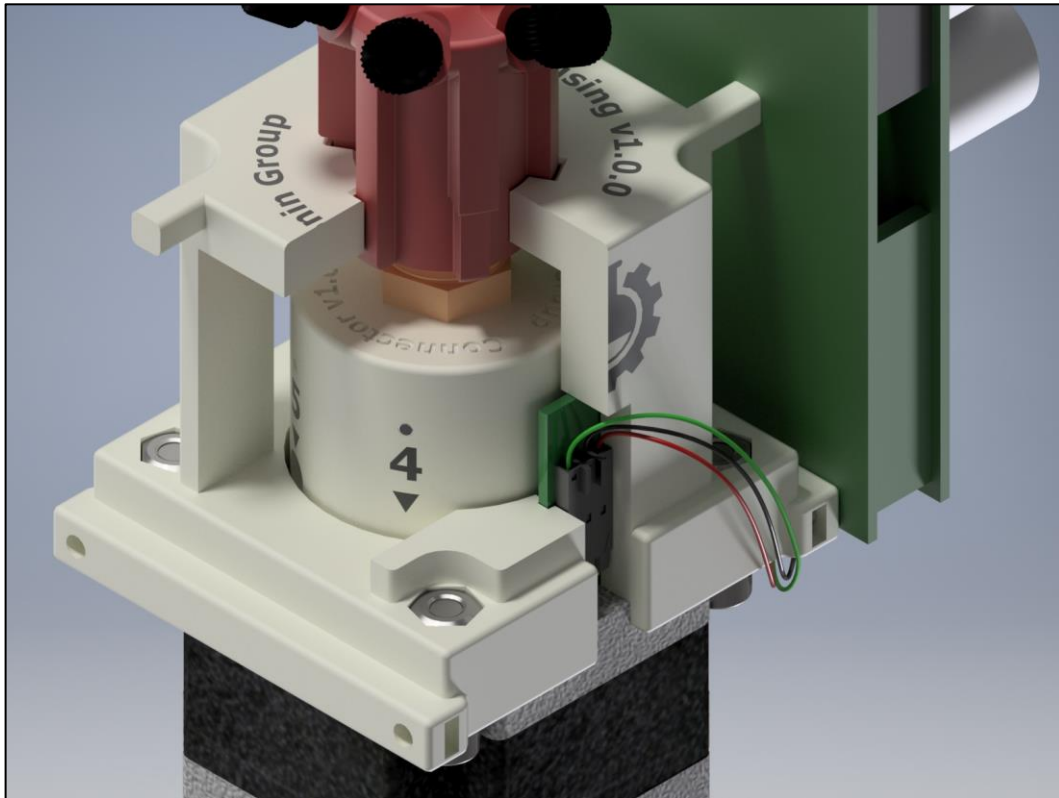


Figure 33: Hall effect sensor breakout board mounted into a valve.

6.1.5 REPLACEMENT OF THE PUMP MOTOR

A common mode of failure encountered often during the early stages of the development of the chemistry was rupture of the syringes (Figure 34). When the pump was dispensing liquid and the flow path got blocked due to precipitation or valve malfunction, or a larger amount of solid accumulated inside the syringe, the NEMA23 motor was strong enough to just break the syringe. This could happen in a benign way such as tearing the glass barrel out of the flange, or in a less benign way such as the glass shattering, sending shards and chemicals flying. In addition to the safety aspect of this bug, the syringes were costly, and exchanging them took time.



Figure 34: Shards of a syringe which was destroyed because a valve failed to switch.

RESULTS AND DISCUSSION

The motor driver chip used provides a sensorless stall detection which should supposedly enable the user to detect a motor overload and subsequently stop the motor. Significant effort was invested into trying to adopt this feature, unfortunately to no avail. After corresponding with the chip manufacturer, it was concluded that the rotation speeds used in this application were too low to allow the measurement principle to work. Additionally, the NEMA23 was so strong that the glass syringe did not constitute a significant load for it. No torque curve for the motor was provided, but a rough estimation based on the limited data available showed that the torque was sufficient to produce multiple times the maximum rated pressure of the syringe.

Thus, it was decided to replace the motor with a smaller NEMA17 model from the same manufacturer. The lead screw dimensions remained the same, so the only changes required were minor alterations of the motor connector. The recess locating the motor was resized to fit the smaller face plate, and the opening for the lead screw was resized to fit the raised circular section of the face plate, in order to further improve the alignment. The protruding feature for mounting the guide rail was shortened so as to sit flush with the bottom surface of the motor, to allow the pump to sit flat on a surface. This required the top screw of the guide rail to be moved into the main body of the connector, which made it harder to access, but as a side effect improved the rigidity of the assembly. Unfortunately, the smaller motor now meant that every syringe would protrude beyond the bottom surface. Various options for avoiding this problem were investigated, but ultimately abandoned. The pumps would usually sit on a shelf, so the protruding syringe wasn't too much of an issue.

Syringes were officially rated to 7 bar, but only tested up to 5 bar. Tests with 25 mL syringes and different backpressure regulators showed that a pump using the NEMA17 motor could produce 5 bar, but would stall against 7 bar backpressure, thus rendering the devices intrinsically safe. Indeed, day to day use later on showed that blockages would lead to a harmless stall rather than a rupture.

With the smaller motor in place, the stall guard was revisited. It was found that at high pumping speeds (corresponding to 50 mL/min with 25 mL syringes, and above) the stall guard could detect a stall correctly, however, at lower speeds the chip would report a stall constantly, thus rendering the feature useless for normal pumping

operation. However, a hard homing algorithm was implemented using the stall guard. The pump would stall itself against its base, thereby establishing a hard home point, then back up a predefined number of steps, and then record the Hall effect sensor reading as new working home position. To protect the syringe and the sensor a thick ring was added to the motor connector to provide a strong surface to stall against. For day to day use, a habit of hard homing the pumps at the beginning of every experiment was adopted to ensure a consistent home position. It was found that this algorithm performed flawlessly.

6.1.6 IMPROVEMENTS TO USABILITY AND AESTHETICS

In the course of the design iterations described above, some shortcomings regarding the overall look and feel of the devices were addressed as well (Figure 38). The initial designs were very minimalistic, which is mainly due to the inappropriate tools used to model them. The older devices looked awkward and blocky and not very professional. Moreover, the sharp edges and pointy corners made them uncomfortable to pick up and handle. Thus, all edges were broken with fillets and corners were rounded where possible. The overall design was simplified, legacy features were removed, and attention was given to making the cleaning of the 3D printed parts as easy as possible. An example of this simplification is the syringe sleeve. Initially, the sleeve was held in position by three locator pegs with angled contact surfaces (Figure 35). This means the entire mechanical strain exerted onto the syringe is ultimately channelled through three tiny features, while the matching grooves in the motor connector were notoriously hard to clean after printing. Yet, the sleeve itself is rotationally symmetric, so there is no point at all in avoiding rotation. Thus, the three locator pegs were replaced with one continuous rim (Figure 35), providing ample contact surface all around. All edges were chamfered, making cleaning of the 3D printed parts easier. In principle, this part could now be turned rather than printed, or if a small draft angle is added, it could be injection moulded.

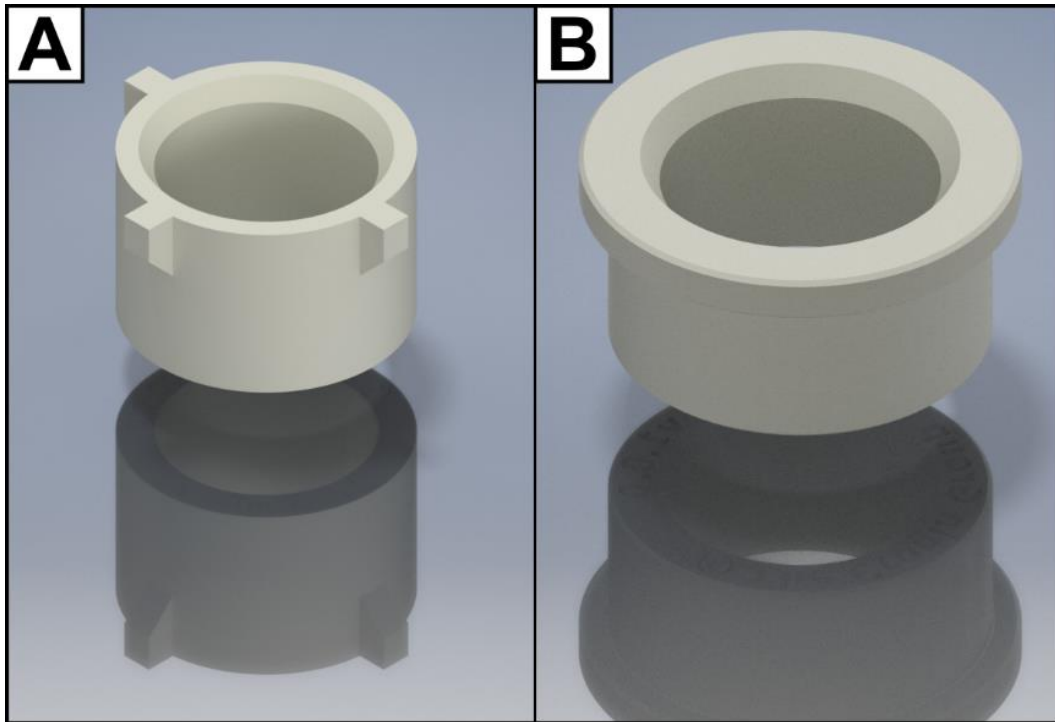


Figure 35: Renderings of the old (A) and the new (B) syringe sleeve.

A logo for the project was designed (Figure 36) and embossed in a different colour (mostly black on transparent yellow) onto the front of the devices.

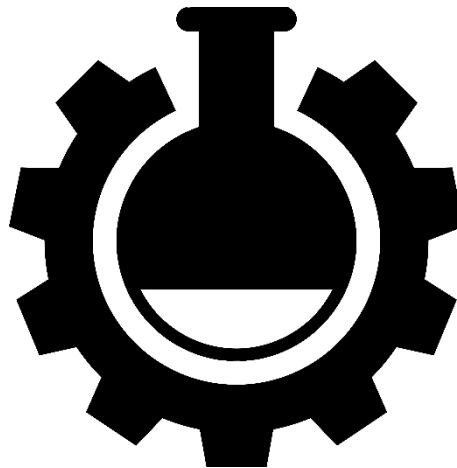


Figure 36: Logo of the Chemputer project.

Swapping the syringe of the pump was previously an inconvenient operation. When the M4 screws holding down the cover plate were undone, the hex nuts would fall out, and in order to fasten the screws again, the nuts had to be inserted into the nut

traps and properly aligned so the screw could bite. To simplify the process, tapered nut traps allowing to press-fit a conventional nut were investigated, but ultimately press-in nuts were used (Figure 37). Those hex nuts feature a knurled rim and can be easily pressed into a circular hole of appropriate size. Replacing the previously used hex nut with press-in nuts made swapping the syringe significantly easier.

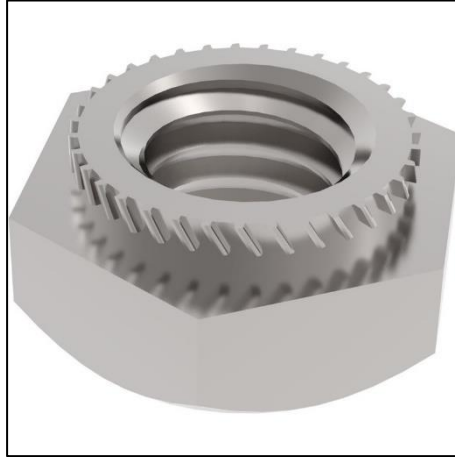


Figure 37: Press-in nut (image reproduced with permission of Accu Limited).

Another shortcoming of the initial design was that it did not provide any means of mounting the control board. In fact, the hardware and the boards were designed independently of each other, with no design specification for fit or compatibility. Previously, the boards were mostly just attached to the devices using zip ties. This produced some problems, because if the blank pins on the back of the board touched any metallic parts such as the motor, the resulting short circuit would often destroy the board. Unfortunately, the boards could not be changed, but mounting points were added to both pump and valve, allowing the existing boards to be screwed securely onto the devices.

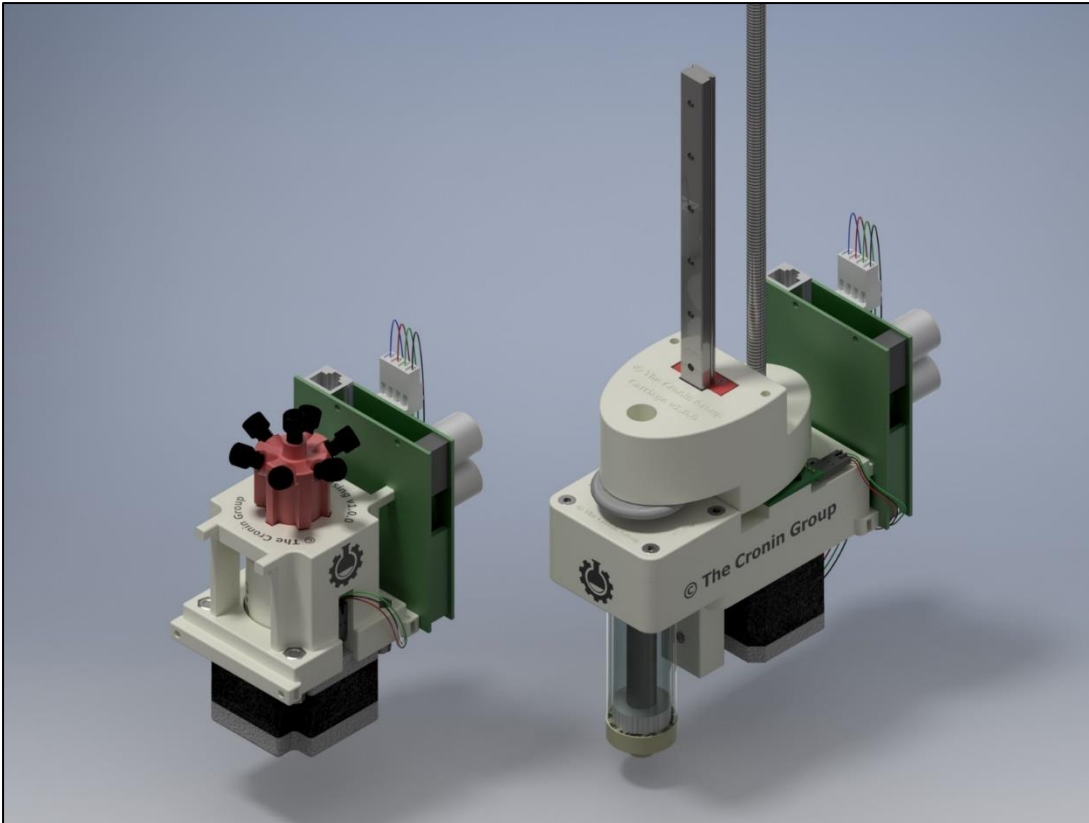


Figure 38: Renderings of the current versions of the pump and valve.

6.1.7 PROJECT DOCUMENTATION

Initially, no formal documentation regarding the earlier iterations of the devices was available. Thus, to assist in future development efforts, the design process was captured in detail.

To that end, 3D models of all third-party components such as the syringes, valve heads, motors, guide rail, etc. were obtained or created and used in full detail assembly files of both the pump and valve. Using Autodesk Inventor's Presentation environment, exploded views of the assemblies were created and annotated to show how to assemble the physical device. Engineering drawings of the 3D printed parts and a Bill of Materials for both devices were compiled as well. Critical features of parts or assemblies were also annotated using the Engineer's Notebook within Inventor.

To explain the overall workflow of printing, assembling, programming and using the devices, an extensive article on the group's internal Wiki was written. The

RESULTS AND DISCUSSION

documentation has since been given to several other group members, who were able to independently build and use pumps and valves.

All engineering drawings can also be found in appendix II.

RESULTS AND DISCUSSION

6.2 CHEMPUTER SETUP

6.2.1 REACTOR MODULE

At the outset of my thesis work, the reactor module was a simple two-necked round-bottomed flask fitted with an air condenser. Heating was achieved *via* a computer controllable hotplate stirrer and a commercially available aluminium heating block. The air condenser was either open to the atmosphere or closed with a rubber septum pierced with a hypodermic needle to ensure pressure equilibration. This precluded the use of air sensitive reagents, an issue that was ultimately addressed by the inception of the inert gas supply system (see chapter 6.2.5).

The side arm of the flask was equipped with another rubber septum fitted with a long hypodermic needle which was in turn connected to the Backbone. While rubber septa and hypodermic needles are commonly used to perform reagent additions in synthetic organic chemistry, it quickly became apparent that for the platform they were an inadequate solution. First, the relatively thin lumen of the needle frequently led to blockages. Secondly, and more importantly, it was found that the stainless steel of the hypodermic needles is corroded heavily if immersed in aggressive reaction media for extended periods of time.

Thus, a solution for directly routing the PTFE tube into the reactor was conceived. Directly pushing the tubing through a rubber septum was investigated, however, a gas tight seal couldn't be established, and the tube often twisted in a way that made it hard to reliably position the tip at the bottom of the reactor. Instead, DURAN® offers the GL 45 and GLS 80 connection systems, which allow the user to connect tubing of various diameters to either GL14 or GL18 threads (Figure 39). Glass adapters from both of those thread types to standard ground glass joints of various sizes are commercially available. This assembly offers a gas tight seal, holds the tube in place reliably, and all wetted surfaces are either glass or PTFE, thus satisfying the requirements regarding chemical stability.

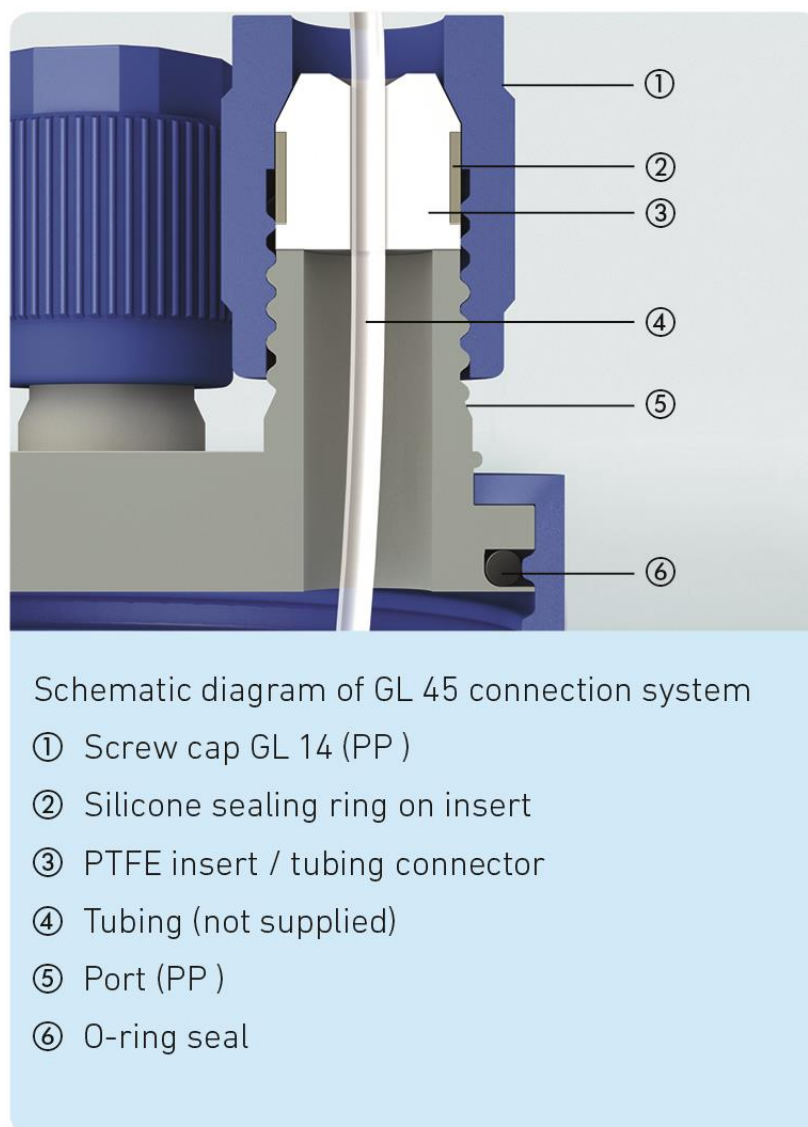


Figure 39: Schematic of DURAN® GL 45 connection system (image reproduced with permission of DWK Life Sciences GmbH).

Based on those findings, and particularly based on the poor performance of the stainless steel needles, the design philosophy for the system was amended to include a strict imperative regarding chemical compatibility: all surfaces that may conceivably come in contact with chemicals should be glass, PTFE, or Viton™. If none of these materials are available, PEEK, PVDF, or PP are acceptable.

As chemical development progressed, it became increasingly clear that the volumes used even within one contiguous synthesis varied significantly. Any reactor vessel had to be dimensioned for the largest volume encountered in the synthesis but should perform equally well for smaller volumes. The classic piece of glassware used

in such a scenario is the pear-shaped flask. The bottom of this type of flask tapers down to a point, which makes it suitable for very small volumes, and also allows reliable quantitative withdrawal of the reactor contents. To the extent of the author's knowledge, no aluminium heating blocks accommodating pear shaped flasks are commercially available at the time of writing, so a custom heating block was designed in house and manufactured by a contractor (Figure 40). The dimensions and shape of pear shaped flasks are guided by DIN 12 383, an old German industry standard. While diverging shapes exist, it was decided to comply with DIN 12 383 to make it easier to replicate the setup. An engineering drawing with all dimensions can be found in appendix II.

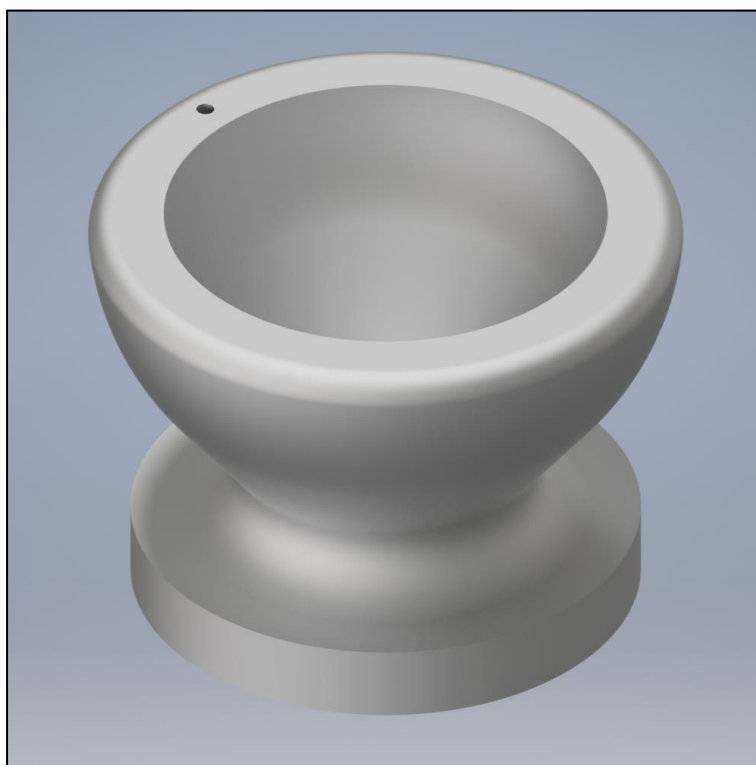


Figure 40: Autodesk Inventor rendering of the custom heating block for 250 mL pear shaped flasks.

For the Grignard step of the Nytol synthesis (see chapter 8.1.4), the previously employed air condenser, a Condensyn by Asynt Ltd. (Figure 41, left) did not perform satisfactorily, due to the high exothermicity of the reaction and the low boiling point of the ether. Thus, it was replaced with a Findenser by Radleys (Figure 41, right),

which successfully contained all ether inside the reactor vessel. For the other syntheses, the slightly cheaper and less bulky CondensSyn was found to be sufficient.

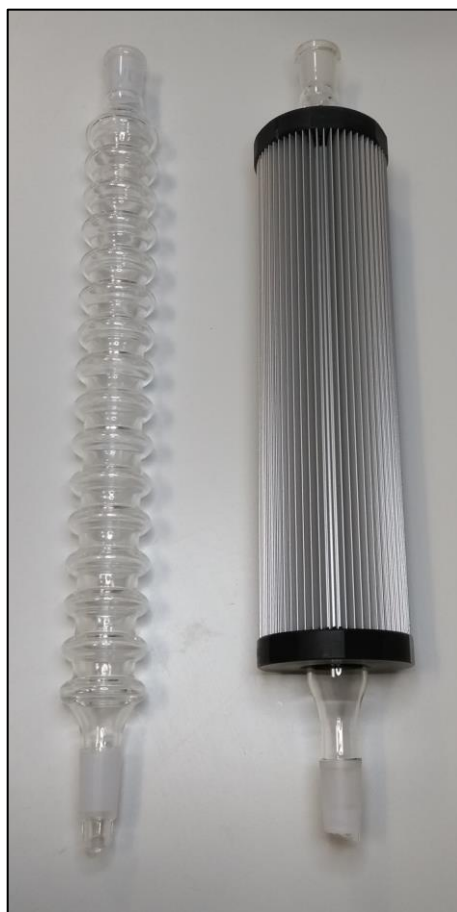


Figure 41: Asynt CondensSyn (left) and Radleys Findenser (right)

6.2.2 EVAPORATION MODULE

Concentrating a product solution *in vacuo* is a pivotal step at the end of nearly every synthetic procedure, and it can be argued that automating this unit operation is the key to unlocking automated multistep syntheses in batch. While solutions for continuous flow exist,¹¹⁵ to the best of the author's knowledge, at the time of writing no other automated batch synthesis platform implements fully automated solvent evaporation. Initially, no evaporation module for the platform existed, and solvent evaporation had to be performed manually.

While automated rotary evaporators are commercially available, at the time of writing they were mostly optimised for continuous operation, allowing the

evaporation of large quantities of solvent. After evaluation of various commercial alternatives, none was found to precisely fit the requirements, especially regarding computer control. Therefore, it was decided to build a custom solution.

The base platform for the development was the IKA RV 10 digital rotary evaporator. This device offers computer control through an RS232 connection, and comes with sufficient documentation to allow us to write bespoke driver software (see chapter 7.2). The original, plastic coated receiver flask was replaced with a custom flask fitted with a threaded glass adapter at the bottom (see Figure 89). This ¼" UNF adapter allowed it to be interfaced with standard tubing using a fluidic union fitting. However, it was found that in this configuration, the vacuum was poor, even if the fitting was plugged with a blanking plug. This issue was solved by introducing a Viton™ O-ring between the glass adapter and the PEEK fitting.

To allow product to be introduced into and withdrawn from the distillation flask, a length of PTFE tubing was routed through the vapour duct. To that end, the tap on the condenser was replaced with a B19 to GL18 adapter, allowing the use of a DURAN GL connector to hold the tubing in a gas-tight way. Leak tests revealed this solution to be sufficiently vacuum tight for this application.

A problem was encountered during the first tests with reaction mixtures. When the internal pressure was decreased for the evaporation, the pressure inside the tube dipping into the distillation flask decreased accordingly. Thus, upon venting the system to ambient pressure, any liquids or oils inside the distillation flask were forced back into the tube. This undesirable behaviour became particularly troublesome if the oil subsequently crystallised, leading to a catastrophic blockage of the tube.

To combat this behaviour, a simple strategy was developed. A small, PTFE coated magnetic stirring bar was attached to the tube using PTFE shrink wrap so that it was positioned inside the B29 ground glass joint of the vapour duct. Then, a strong neodymium magnet was mounted above the joint so as to attract the magnetic stirring bar, and thus lift the tube high enough to avoid material being pushed into the tube. Figure 42 shows an image of the assembly. In order to withdraw material from the evaporator, the computer controllable lift of the evaporator was simply

lowered down. This moved the stirring bar away from the magnet, and the tube dropped to the lowest point of the distillation flask, allowing quantitative aspiration of the contents.



Figure 42: Detail view of the inlet tube. Note the neodymium magnet mounted above the flask, and the magnetic stirring bar attached to the tube.

Another consideration was the ability to flush out the entirety of the product with small amounts of solvent. Removing all solvent from a mixture will in most cases deposit the desired product more or less evenly over the entire inner surface of the flask. With a round bottomed flask, a smaller amount of solvent will only wash a ring out of the deposit due to the curved surface (Figure 43, A and B). Once again, the humble pear-shaped flask proved to be the ideal solution. When aligned in such a way that the straight lower section of the flask was horizontal, a small amount of solvent would cover the entire straight section (Figure 43, C). When the flask was rotated along its axis, the solvent would wash over most of the internal surface area, dissolving the maximum possible amount of material (Figure 43, D). It was found that the flask allowed volumes as low as 5-10 mL of solvent to sufficiently remove solid residue.

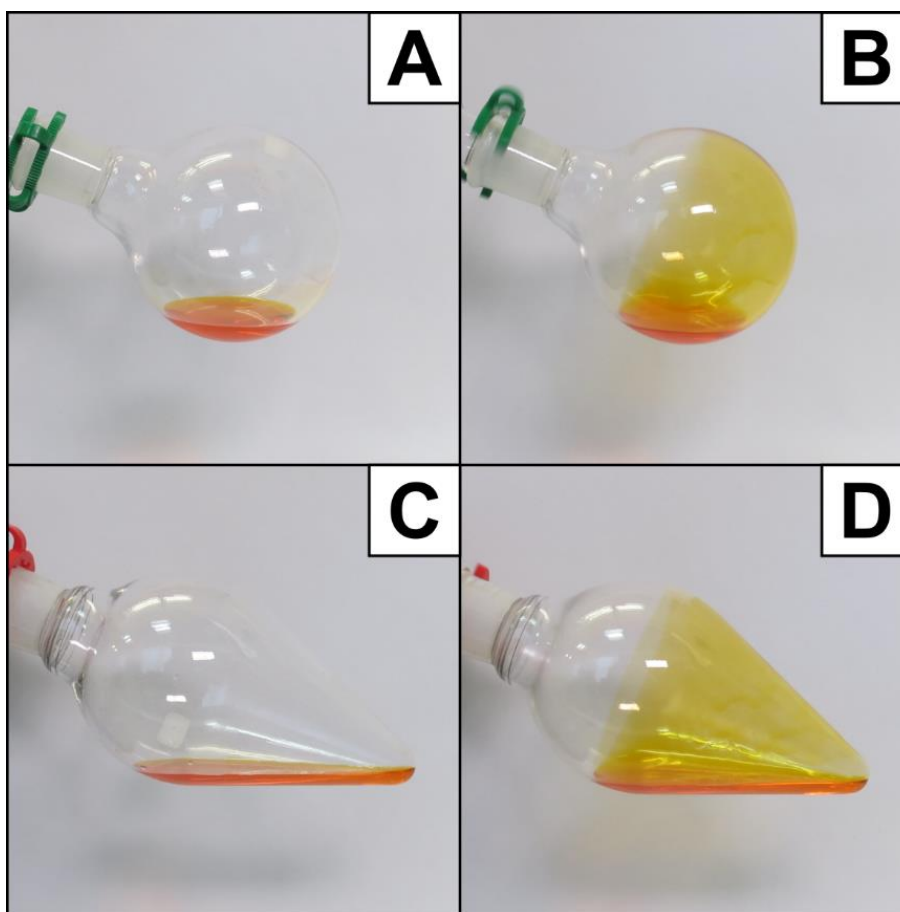


Figure 43: 10 mL of acetone dyed with Sudan I in a 250 mL flask on a rotary evaporator. A) round bottomed flask, standing still. B) round bottomed flask, rotating at 100 rpm. Some of the solvent is dragged up the wall, but the bulk pools at the bottom, only reaching a small fraction of the internal surface. C) pear shaped flask standing still, levelled so the straight part is horizontal. D) pear shaped flask, rotating at 100 rpm. Note how the solvent washes over most of the internal surface.

As vacuum pump the Vacuubrand MD1C vario was chosen; a compact, powerful, variable speed diaphragm pump. It is fitted with the fully computer controllable CVC3000 vacuum controller. In principle, it features an automated boiling point finder which should make it possible to evaporate any solvent or mixture autonomously. However, after extensive experimentation with various pure solvents as well as reaction mixtures, this feature could not be successfully utilised. Ultimately, empirically obtained values for the pressures and times were hard-coded for every operation.

6.2.3 AUTOMATED LIQUID/LIQUID EXTRACTOR (ALLEX)

The liquid/liquid separation, being one of the most common isolation techniques, was also the most challenging task to automate in a robust fashion. In continuous flow, membrane technologies can be utilised.¹¹⁶ The biphasic mixture is flowed past a microporous PTFE membrane, and as long as the pressure difference across the membrane is appropriate, the organic phase will permeate the membrane, while the aqueous phase will be retained.

However, this particular solution was not anticipated to work in the platform, for a number of reasons. First, it was questionable if the pumps could reliably deliver the pressure required to operate such a device. Then, under millifluidic conditions, biphasic mixtures either segregate into a slug flow, or travel along as a parallel flow.¹¹⁷ Either way, both organic and aqueous phase travel alongside each other, at a relatively stable volume ratio. However, in the system at hand it was observed that most biphasic mixtures settle down quickly once they reach the syringe of the pump, so rather than pushing a micromixed biphasic mixture through the separator, a large bolus of aqueous phase should be expected, followed by an equally large bolus of organic phase, and so forth. It was unknown how a flow separator would react to such an operational mode. Lastly, it was questionable how a separator optimised for continuous flow would cope with small, discrete volumes as is the case in the platform. All those concerns are entirely hypothetical; however, the substantial cost of commercially available flow membrane separators precluded any practical testing.

Another method similar to membrane separators is the use of hydrophobic frits made from materials such as UHDPE or PTFE. Unlike microporous membranes, those macroporous frits allow organic liquids to pass through under gravity alone, while retaining the aqueous layer. They are commercially available, and commonly used in analytical sample preparation. However, the most common design, a syringe barrel with a frit at the bottom,¹¹⁸ necessarily only works with organic solvents that are heavier than water. Fortunately, Biotage[®] supplies a product named Universal Phase Separator. It consists of a vertical, tubular hydrophobic frit situated inside a larger plastic tube with a Luer fitting at the bottom (Figure 44). The biphasic mixture is introduced into the space between the two concentric tubes. The organic phase

permeates into the inner tube under gravity and can be collected through the Luer fitting. This geometry means both organic phases heavier than as well as lighter than water can be separated.



Figure 44: The Universal Phase Separator (image reproduced with permission of Biotage AB).

In order to interface this device with the platform, a short length of PTFE tubing was fitted with a $\frac{1}{4}$ " UNF to Luer lock adapter and placed atop a three-necked round bottom flask using a DURAN connector and GL18 to B19 adapter. Another tube reaching the bottom of the flask was secured using a second DURAN connector and glass adapter. This tube was connected to the Backbone and allowed emptying of the permeate. The Universal Phase Separator was connected to the Luer lock adapter and supported by a clamp. A tube leading to the Backbone was secured to the inner tube of the Universal Phase Separator so that it reached the lowest point, allowing the introduction of biphasic mixture, and withdrawal of aqueous retentate. Figure 45 shows an image of the setup.



Figure 45: Universal Phase Separator setup.

This setup was tested extensively during the development of the Williamson ether synthesis in the Nytol synthesis (see chapter 8.1.3). Unfortunately, it was found to perform rather poorly. The separation was slow, and small amounts of organic phase always remained in the separator. Moreover, after the first separation, the separator often started to leak on subsequent separations, allowing copious amounts of aqueous layers through. The reason for this poor performance could never be determined, even though Biotage kindly provided us with replacements, and gave us advice on how to improve the performance. It was thought the chosen separators, whilst designed for single use, would be able to sustain multiple identical uses in rapid succession. This would be required, for example, during extraction of an

aqueous layer multiple times, however the units failed during this process. Furthermore, the separator sometimes even broke down on the very first separation. Commonly, the mode of failure would be a leakage of aqueous layer through the frit. Initially, small amounts of water were observed in the collected organic phase. Then, after some minutes, the aqueous phase would flow almost freely through the separator. Ultimately, the approach was abandoned in favour of a more conservative solution.

The way a human chemist discerns a phase boundary in a separatory funnel is by visual inspection. Thus, it was decided to attempt to use machine vision to automate liquid/liquid separation. This approach had several advantages. First, it is agnostic to which layer is organic and which is aqueous, as long as they are visually different, a separation can be performed. Second, it was a non-invasive method that did not require any sensors to be in contact with potentially aggressive media, thereby increasing reliability. Additionally, a camera-based solution was hoped to not only find the phase boundary and perform the separation, but also determine if and when the settling process was complete. Based on the densities and compositions of the two layers, the time it takes for a biphasic mixture to settle into two clearly delineated layers varies quite significantly. If the system could dynamically determine the end of the settling process, the overall operation would become both faster and more reliable.

There is literature precedent for the use of coloured floaters¹¹⁹ as well as direct recognition of the boundary based on edge detection.¹²⁰ Additionally, the group had experience with classifiers for image recognition¹²¹, so it was reasoned that a classifier could be trained to directly spot phase boundaries in an image.

To test those methods, a custom-made separatory funnel was designed for the application. The initial module was based on a standard 100 mL separatory funnel. The tap and bottom outlet were replaced with a ¼" UNF threaded glass adapter to allow convenient interfacing with the PTFE tubing and the Backbone. Two B14 side arms were added, and an inlet tube was mounted to one of the side arms via the usual DURAN adapter assembly. At first, the coloured floater approach was investigated. Green polypropylene balls (5 mm diameter) were obtained from The

Precision Plastic Ball Company Ltd. (UK). A simple Python script using OpenCV allowed reliable detection of the green ball by applying a filter removing everything outside a narrow colour range and detecting closed contours in the resulting image. The bottom of the separatory funnel was covered with brown electrical tape, so the green ball disappeared from the camera's view once it dropped below the tape.

After giving the biphasic mixture sufficient time to settle, a small volume (1 mL) was withdrawn from the bottom outlet. Then, a frame was captured, and the image recognition function was run on it. If a green ball was detected, another portion was withdrawn, and another frame was captured, and so forth. Once the green ball disappeared, the residual dead volume of the separator and tubing was withdrawn, and the upper layer was transferred to the target vessel. This algorithm was relatively inefficient, but the exceedingly simple implementation (just over 30 lines of Python code) developed by Dr. Jaroslav Granda yielded a working prototype ready for testing.

The algorithm worked well in a test system consisting of pure water and diethyl ether or ethyl acetate. Unfortunately, it quickly reached its limits when confronted with a real reaction mixture. First, when the mixture was strongly coloured or cloudy, the green ball would often become invisible if it floated at the far side of the separatory funnel. Since the horizontal position of the floater is essentially random, this proved to be a big problem. Even strong lighting from various angles did not resolve the issue. Another unexpected complication was encountered during the development of the Williamson step of the Nytol synthesis (see chapter 8.1.3). The original protocol used p-xylene as reaction solvent, which has a higher density than the polypropylene ball, so the ball would float on top of the organic layer instead of between the aqueous and organic. Even though the solvent was later changed to toluene, this problem clearly demonstrated the limitations of the approach. Additionally, while the polypropylene worked in principle for organic solvents lighter than water, no workable solution for solvents heavier than water was available at that point. Considering all of these issues, the approach was eventually abandoned.

Next, the line of work was bifurcated. While I proceeded to investigate sensor-based solutions (*vide infra*), Dr. Granda set out to employ a deep convolutional neural

network to classify images of the separatory funnel into “phase boundary present” and “phase boundary absent”. This method was first reported by Krizhevsky, Sutskever and Hinton in 2012¹²² and at the time of writing their paper had been cited over 25,000 times. Since sensor-based approaches were being investigated at the same time (*vide infra*), nearly two and a half thousand images were aggregated as training set for the classifier. Unfortunately, it turned out that in a practical application there are a number of difficult and ambiguous cases which in some instances make it hard even for a human to discern a phase boundary, or the absence thereof. Figure 46 shows some examples. Case A) shows deposits left on the inside of the separator. In truth, the vessel is empty, but it looks like there is a phase boundary and may lead to false positives. Case B) illustrates the opposite, in this case both phases are strongly coloured, and even with strong backlight they look like just one phase. Visual inspection of the separatory funnel at this point did however confirm that there were in fact two layers present. Case C) shows poor settling leading to clumps of emulsion clinging to the separator. This makes it very hard to judge if there is a boundary present. Case D) illustrates a rather common case where the volume of the upper layer is significantly smaller than the volume of the bottom layer. This may confuse a neural net trained on cases where the volumes were more equal. Cases E) to H) are taken from one single separation. First, the initial volume was too large for the vessel, leading to an atypical appearance. After withdrawing some of the lower phase, it became apparent that there was some emulsion in between the layers, leading to what looks like two boundaries. Case G) is an advanced stage of the separation, with all of the bottom layer withdrawn, and only the top layer and the emulsion remaining in the separatory funnel. Now it looks like a regular case, with one boundary. After withdrawing all the emulsion as well, only one phase remains, but with some specks of emulsion still clinging to the walls (case H). Beyond those irregularities and complications, a wide variety of colours was observed for both phases, which further complicated the matter.

Admittedly, the frames selected for Figure 46 were extreme examples, and by tuning the chemistry of the extractions matters were improved a bit, but the difficulties encountered discouraged further pursuit of a computer vision based approach.

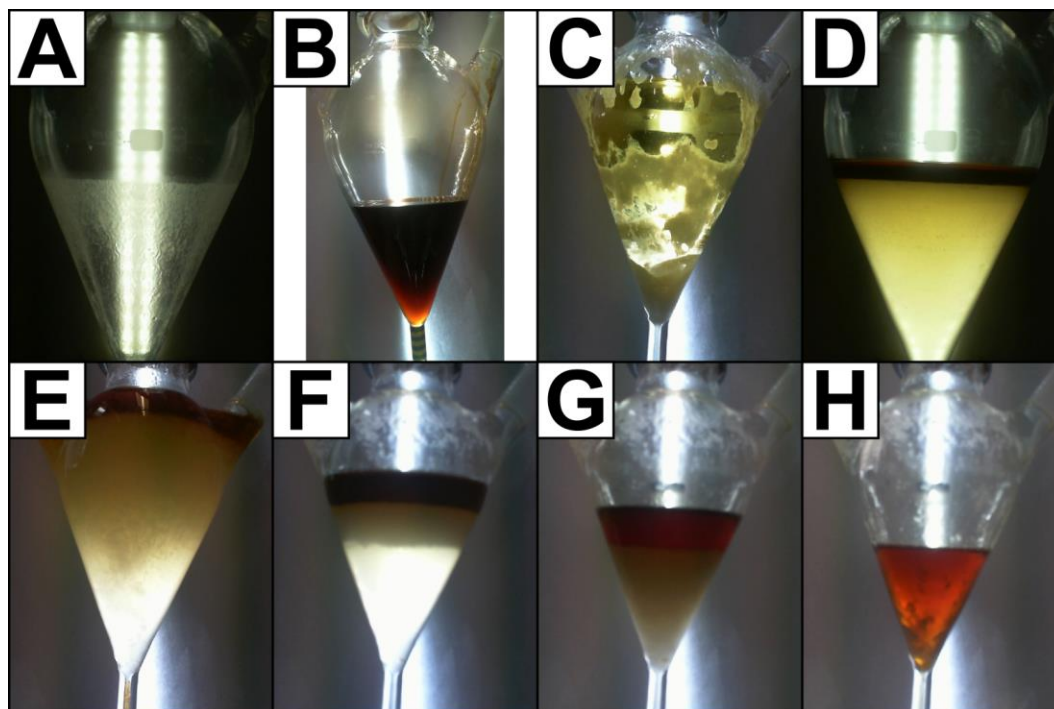


Figure 46: Examples of difficult or ambiguous frames for phase boundary detection. A) false positive due to deposits on the glass. B) false negative due to strongly and similarly coloured phases. Visual inspection of the separator did indeed show two layers, but the poor image quality and angle of view obscure the fact. C) fluffy emulsion deposits due to poor phase separation. D) small organic volume. E) overfilling. F) poor separation leading to two boundaries. G) same separation a bit later, organic phase and emulsion are visible. H) same separation, only organic phase remains, but with emulsion clinging to the side.

As the image recognition proved to be more challenging than anticipated, attention was shifted to sensor-based approaches. Literature precedent for conductivity sensors used in a very similar application was found,¹⁰³ however, given prior adverse experiences with the stainless steel needles, contactless measurement principles were investigated first.

Thus, a fluid and bubble sensor for 1/8" tubing manufactured by TT Electronics plc. was sourced. This sensor was originally designed for medical applications such as monitoring haemodialysis machines or IV drips and consists of an LED and a phototransistor mounted in a plastic housing moulded to fit around a tube with 1/8" outer diameter. Figure 47 shows a photo of the sensor with a short length of tubing inserted for illustrative purposes. The idea was to clip the sensor onto the tube

between the bottom outlet and the Backbone, withdraw liquid from the separator in a stepwise fashion, and collect measurements after each step similar to the algorithm described above.

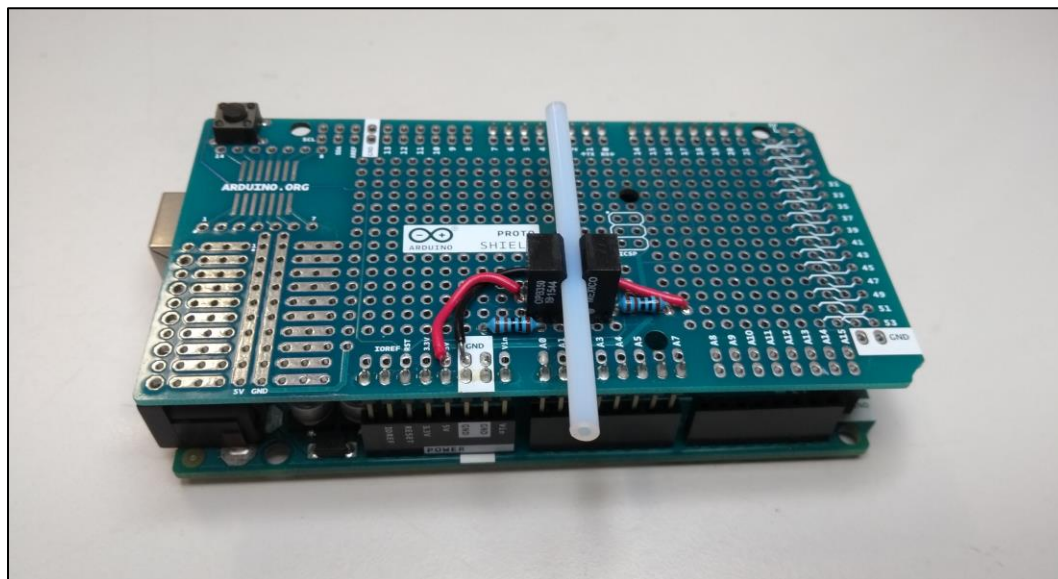


Figure 47: Bubble sensor mounted on an Arduino prototyping shield for testing.

A quick-and-dirty voltage divider circuit implemented on an Arduino prototyping shield allowed us to quickly conduct some initial experiments. First attempts using pure water and organic solvents yielded small, but consistent differences in the sensor reading. Encouraged by those findings, real reaction mixtures obtained from the Grignard or the Williamson step of the Nytol synthesis (see chapters 8.1.4 and 8.1.3) were tested. Unfortunately, those mixtures were often found to be turbid due to particulates or tiny droplets suspended in the bulk liquid. Those mixtures yielded almost random sensor readings. It was reasoned that in the case of a turbid solution, light scattering becomes the dominant effect determining the sensor reading, and the absorption response gets blotted out. Investigating more robust circuitry to smooth the response and improve performance was briefly considered, but ultimately optical sensing for phase boundary detection was abandoned.

Next, capacitive sensors were investigated. In the briefest of descriptions, a capacitor consists of two conductive surfaces separated by an insulating layer. A capacitor can hold an electric charge when a voltage is applied between the two conductive

surfaces. The physics of capacitors is beyond the scope of this chapter, suffice to say that the amount of charge (Q) a specific capacitor can hold is proportional to the voltage (V) across it, and a specific property called capacitance (C) which can be measured:

$$Q = C * V \quad \text{Equation 1}$$

Ultimately, the capacitance of a capacitor is a function of three parameters¹²³: distance between the plates (d), area of the plates (A), and dielectric constant of the material between the plates (ϵ_r):

$$C = f(d, A, \epsilon_r) \quad \text{Equation 2}$$

The dielectric constant is a material property and a dimensionless number. For the application in question, the geometry of the capacitor (d and A) is kept constant, while the material and therefore the dielectric constant changes, which allows us to infer the material by measuring the capacitance. This approach was seen as particularly promising since the dielectric constants of most common organic solvents are about one order of magnitude smaller than that of water:

Table 1: Dielectric constants of common laboratory solvents at 20°C¹²⁴

Solvent	Dielectric constant
Water	80.37
Dichloromethane	9.08
Diethyl ether	4.34
Ethyl acetate (25°C)	6.02

Capacitive sensing is widely used in various industries. Position and distance sensors, liquid and solid level sensors, certain touchscreen technologies, and many more applications use capacitive sensing. Unfortunately, finding or building appropriate instrumentation for the application in question proved to be more difficult than anticipated.

Most cheap commercially available capacitive sensors operate as switches, i.e. an output lead changes its state if a predetermined capacity threshold is crossed. This mode proved unsuitable for determining phase transitions, as the threshold would have to be tuned for every new separation. True analog capacitive sensors were prohibitively expensive. Custom built sensors were devised, but a geometry that allowed laminar flow through an appropriate diameter and still provided useful readings could not be achieved.

Since at this point a working solution for the liquid/liquid separation was the major bottleneck of the entire project, a decision was made against investing more time into developing a capacitive sensor, and instead settled for a conductivity sensor.

The two major design considerations for a flow conductivity sensor for the application in question were the following. First, it was required to be easily interfaced with standard 1/8" tubing. Second, the dead volume should be minimal. Additionally, time was a consideration as well since the project was stalled until a working solution was found. Unfortunately, extensive research did not yield a single suitable, commercially available sensor. Thus, a custom solution was built (Figure 48).

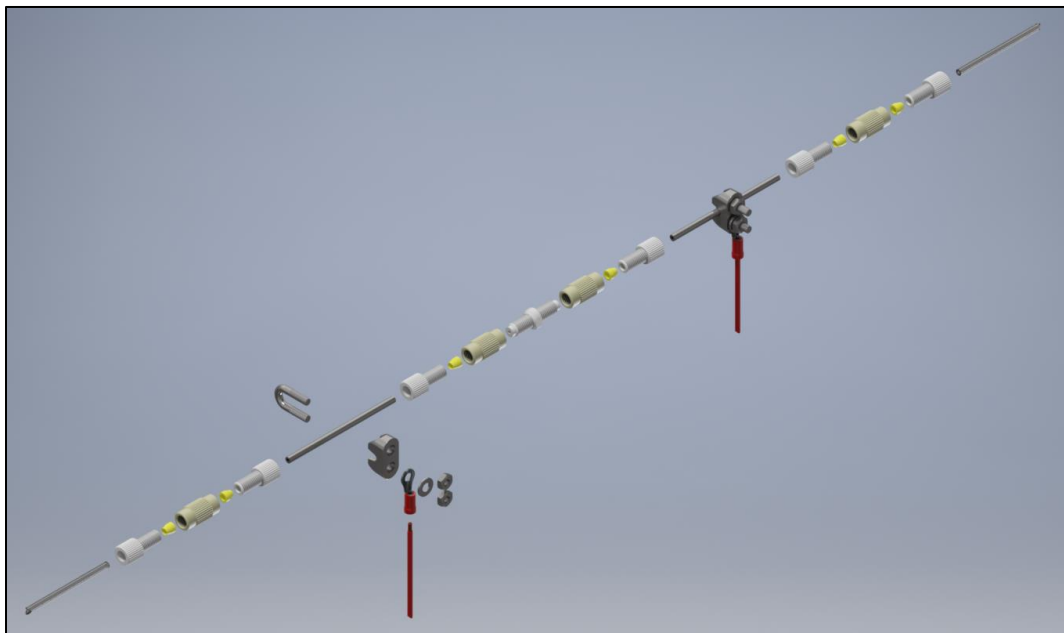


Figure 48: Autodesk Inventor rendering of an exploded view of the conductivity sensor. Two pieces of 1/8" O.D. stainless steel tubing are separated by insulating fluidic unions. Wire rope grips connect the steel tubes to wires. The phase is determined by measuring the resistance across the two steel tubes.

To minimise cost and lead time for a working prototype, it was decided to assemble a sensor from off-the-shelf millifluidic supplies available at the time. The sensor consisted of two lengths of stainless steel tubing separated by a non-conductive spacer. Stainless steel was a controversial choice due to chemical compatibility concerns. However, it was reasoned that, unlike the hypodermic needles in the reactor, the sensor would only be in contact with aggressive media for a short time, and only at room temperature. Thus, corrosion was expected to be less of an issue. Transition metal ion contaminants leaching from the stainless steel were a concern as well, but it was judged unlikely that such contamination would interfere with the chemistry investigated at the time. Based on those assessments the shortcomings of stainless steel were weighed against the low price and ready availability and it was decided to test the setup with stainless steel tubes. The reasoning at this point was that a working prototype was more desirable than a perfect solution, and due to the modular nature of the setup, more suitable sensors could be introduced at any time during further development.

To ensure a reliable electric connection to the tube, wire rope grips were employed in conjunction with crimp-on ring terminals for the wires. A simple voltage divider circuit on an Arduino prototyping shield completed the setup. The sensor assembly could be connected directly to the bottom of the separating funnel *via* the threaded glass adapter. The size of the separatory funnel was increased to 250 mL.

This admittedly crude solution performed surprisingly well and was used extensively for both the Nytol and the Sildenafil synthesis. Initially, irregularities due to deposits inside the sensor assembly were sometimes encountered, but those were easily resolved by adhering to a strict cleaning regime after every separation. The withdraw-and-measure algorithm was employed again since it was proven to work well, and also because the structure of the control software made any more sophisticated approach relatively difficult to implement. However, it was found that moving the bulk of the lower layer to the destination vessel based on an estimation of its volume was a simple way to greatly decrease the overall separation time, so a decision was made against investing even more time to implement an intrinsically more efficient algorithm.

With a working liquid/liquid separator in hand, the development of the syntheses could finally progress. However, another shortcoming of the system was soon noticed. The mixing during extraction processes was achieved by pumping both phases into a dedicated round bottomed flask, and magnetically stirring the contents. Unfortunately, the mixing efficiency was decidedly unsatisfactory, leading to poor extraction efficiency. Also, having to move large quantities to the holding flask and back for every extraction prolonged the operation significantly. To address this problem, an overhead stirrer was introduced into the separator. To that end, the central B29 joint was replaced with a wider B45 joint to allow the insertion of a stirring rod. This improvement not only increased extraction efficiency, but also allowed us to improve the settling efficiency by slowly stirring the contents during settling of the layers. This slow stirring broke up foamy emulsions and detached droplets stuck to the glass wall, similar to how a human chemist would gently swirl a separating funnel to improve settling.

A full description of the liquid/liquid separation module in its most current state can be found in chapter 10.2.

6.2.4 FILTRATION MODULE

At the outset of my thesis, filtration was performed using a sintered glass Büchner funnel with a B19 ground glass joint at the top and bottom, respectively, and a B19 sidearm (Figure 49). The Büchner funnel was placed onto a round bottomed flask, the sidearms of the flask were fitted with rubber septa, and a stainless steel hypodermic needle was inserted through one of the septa to connect the flask to the Backbone. The two ground glass joints at the top of the Büchner funnel were fitted with rubber septa as well, and two stainless steel hypodermic needles were inserted. The needle in the sidearm was connected to the Backbone to allow material to be introduced, while the needle in the central port was open to the atmosphere for pressure equilibration.



Figure 49: Initial filtration module.

Aside from the material compatibility considerations mentioned in chapter 6.2.1, this setup performed rather poorly, as it was very hard to establish a gas-tight seal for the round bottomed flask, so withdrawing a volume from the flask would not necessarily produce enough vacuum to facilitate an efficient filtration. Additionally, the dead volume under the filter frit was considerable. It was already found in previous experiments that for any operation that led to crystallisation inside the filter, the dead volume underneath the frit had to be filled with solvent to avoid solution leaking through the frit under gravity, which would in turn lead to loss of yield at best, or blockage of the filter at worst. With the comparatively large flask, this filling operation was very slow and inefficient.

Thus, a new Büchner funnel was designed (Figure 50, left). The overall concept was retained, but instead of placing it onto a flask, a threaded glass adapter was fused directly to the bottom outlet, allowing a direct interface with the Backbone, and minimising the dead volume. The adapter was bent at a 90° angle to allow the vessel to be suspended above a hotplate stirrer and be magnetically stirred. The top B19 joint was replaced with a wider B29 joint to allow convenient emptying of solids from the vessel, while the sidearm was reduced to B14 and fitted with a DURAN adapter.



Figure 50: Custom made Büchner funnel. Left: unjacketed. Right: jacketed.

This new filter performed exceptionally well for the hydrochloride precipitation in the Nytol synthesis (see chapter 8.1.5). The small dead volume below the filter frit (10 mL, including the tubing) could conveniently be filled with diethyl ether to avoid any hydrochloride crystallising inside the frit and potentially blocking the filter. The filtration could be conducted by simply withdrawing liquid from the bottom port.

However, some problems were encountered during the development of the Rufinamide synthesis (see chapter 8.2). The last step of the synthesis produced the title compound as precipitate and transferring the resulting suspension to the filter module led to blockage of the fluidic system. Unfortunately, this step required heating of the reaction mixture for several hours, so it could not be performed inside the filter vessel.

It was reasoned that being able to heat and cool the contents of the filter would be generally desirable, since this would not only facilitate reactions performed inside the filter vessel, but also enable recrystallisations, which are a commonly employed purification technique. Thus, a jacketed version of the filter was designed (Figure 50, right) which could be used in conjunction with a recirculation chiller. Due to the added bulk of the jacket, magnetic stirring was not an option anymore, so an overhead stirrer had to be employed. Since the B29 joint was henceforth occupied by the stirrer gland, a second B14 sidearm was added to allow pressure equilibration, and the introduction of inert gas (see also chapter 6.2.5).

The jacketed Büchner funnel performed well for the Rufinamide synthesis and the recrystallisation of the crude Nytol (see chapters 8.2 and 8.1.8) and proved to be invaluable for the Sildenafil synthesis (chapter 8.3).

Initially, precipitate collected in the filter was only dried superficially by sucking air or argon through the filter cake using the Backbone pump. Full drying was performed manually and offline. However, the Sildenafil synthesis required proper drying of some intermediates, so ways to vacuum dry the filter cake were investigated. After some experimentation, a six-way valve (chapter 6.1) was used to switch the bottom outlet of the filter between the Backbone and vacuum. To that end, the filter outlet was connected to the central inlet of the valve with a short length of tubing. One

outlet of the valve was then connected to the Backbone, while another outlet was connected to a Woulff bottle which was connected to the laboratory vacuum. During normal operation, the six-way valve would connect the filter bottom to the Backbone, allowing the introduction and withdrawal of liquid. For drying the filter cake, the valve would be switched to vacuum, and argon would be sucked through the filter cake. Without another actuator to close off the inert gas supply, the pressure inside the filter would not drop significantly, but the stream of dry gas, especially in conjunction with external heating through the jacket, was found to be enough for efficiently drying the product.

6.2.5 INERT GAS SYSTEM

At the beginning of my PhD, the entire platform was essentially open to the atmosphere. However, many useful reactions in organic chemistry require dry or oxygen-free conditions (or both), so an inert gas manifold was highly desirable.

The classic way of handling air-sensitive reagents is the Schlenk line. This piece of glassware consists of two manifolds, one for vacuum, the other for inert gas. They are connected by three-way valves, allowing the experimenter to switch a vessel connected to the Schlenk line between the two states. Automating this setup was judged to be feasible, but too complicated a task to be tackled at that point.

Instead, for a simpler setup was devised (Figure 91, left, on page 225). The task of supplying inert gas to the system could be subdivided into two distinct parts. First, the reactor and the filtration unit required inert gas input, as well as a means of pressure equilibration to cope with varying temperatures and internal volumes. Second, a variable number of reagent bottles required an inert gas input. As those bottles were not expected to experience significant temperature fluctuations or reactions taking place inside, and volume would only ever be withdrawn from them, a simple non-return valve to prevent vapours diffusing back into the manifold would be sufficient.

Based on this assessment, a simple, two-tiered inert gas system was designed. A flow regulator was mounted directly into the argon tap of the fume hood, allowing the

experimenter to dial in an appropriate argon flow. This regulator was connected to the first tier of the system, a pneumatic manifold with ten outlets. Every outlet was fitted with a pneumatic check valve to avoid contamination with air. Those check valves could then be either plugged with blanking plugs if not required or fitted with a tubing assembly consisting of a chemically resistant polypropylene check valve and a length of Viton™ tubing (Figure 91, right). This tubing assembly would then be connected to the reagent storage bottle (see chapter 6.2.6) via a DURAN GL 45 connection system bottle top.

After the manifold and another pneumatic check valve, the second tier consisted of 8 mm I.D. PVC tubing connected to the reactor and filter, followed by an overpressure bubbler (Figure 91, centre).

This simple, inexpensive setup reliably kept the entire platform under inert gas and enabled us to perform a number of air-sensitive reactions under full automation.

6.2.6 REAGENT STORAGE

The reagents and solvents required for a given synthesis were stored in GL45 bottles (Figure 92 on page 227) of varying sizes, fitted with DURAN GL 45 connection system bottle tops (Figure 39). A length of PTFE tubing leading to the Backbone was inserted through one of the two ports. The other port was left open to the atmosphere where appropriate. For reagents that required an inert gas blanket, a short (approx. 2 cm) length of PTFE tubing was inserted into the other port through a second DURAN adapter, and the 3 mm I.D. Viton™ tube coming from the inert gas system (see chapter 6.2.5) was pushed onto the 3.2 mm O.D. PTFE tube, supplying argon to the bottle.

The slight overpressure inside the reagent bottles led to some unexpected difficulties. Because of the check valves, bottles containing chemicals with higher vapour pressure would consequently have a higher internal pressure as a result. When liquid was aspirated from any given bottle, the internal pressure inside the syringe of the pump would equilibrate with the internal pressure of the bottle. Since there are almost inevitably small gas bubbles inside the syringe at all times, this

means that when the valve switched to another position and passed over outlets connected to bottles with lower internal pressure, some liquid would be pressed into that outlet. This led to cross-contamination, and in some cases even to blockages.

To avoid chemicals getting pushed back into the wrong tubes, all Backbone outlets connected to reagent bottles were fitted with non-return valves (Figure 51). The model used consisted of a PEEK body and a Viton™ membrane and was fitted with a male ¼" UNF thread on one end and a female ¼" UNF thread on the other end. The bore size was specified as 0.060" or 1.5 mm. Those check valves reliably solved the cross-contamination issue and reduced blockage-related issues. The increased flow resistance required some of the lower boiling solvents to be aspirated at a reduced rate to avoid cavitation, but this was never an issue.

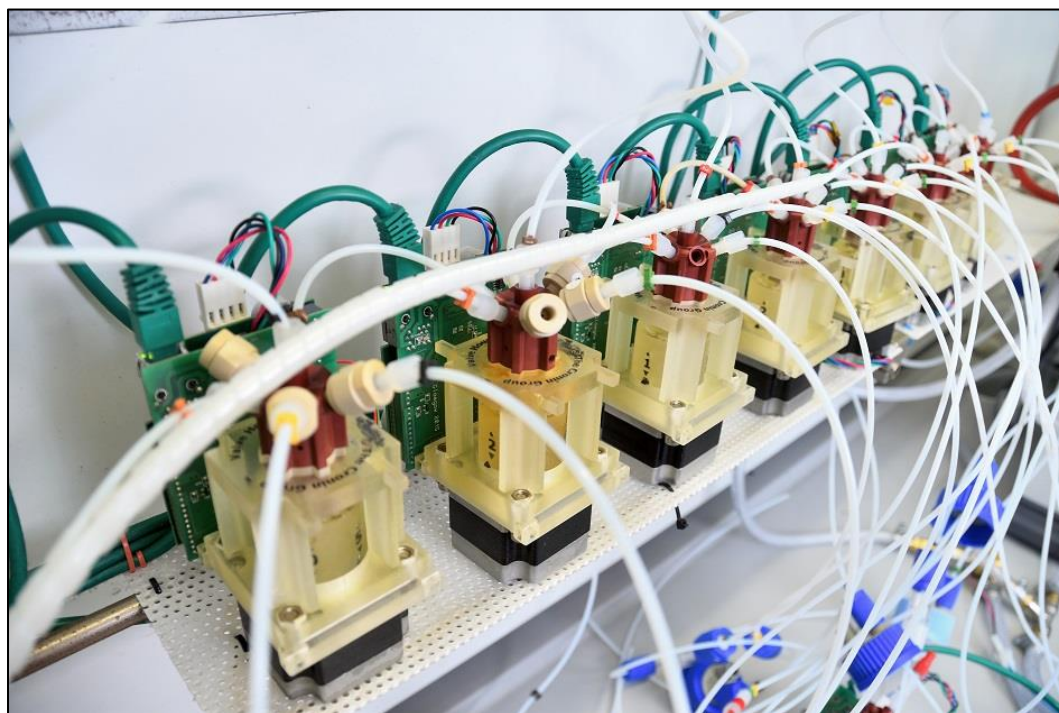


Figure 51: Photo of the Backbone showing the beige non-return valves screwed into some ports of the red six-way valves.

6.2.7 OTHER MODULES

During development of the abovementioned major modules, a number of smaller add-ons were developed as well. One noteworthy development is the introduction of a solenoid valve into the flow path of the recirculation chiller. Initially, the platform

for the Nytol development used its recirculation chiller exclusively for the condenser of the rotary evaporator. When the Sildenafil platform was constructed, it was fitted with a SnowStorm cooling module as well as a jacketed filter, so it was decided to daisy-chain those two modules and use tap water for the rotary evaporator. However, it quickly became apparent that the daisy chain had the significant disadvantage that both the reactor and the filtration module always shared a temperature. This proved to be an issue for the quench of the first step of the Sildenafil synthesis (see chapter 8.3). The filter had to be kept near zero degrees to avoid hydrolysis of the chlorosulfonyl moiety (see Figure 83 on page 199). However, with the reactor near zero degrees as well, the viscosity of the reaction mixture was increased to a point where pumping became near impossible. In order to allow us to heat or cool the two units selectively, a three-way solenoid valve was introduced into the flow path. The driving circuit was implemented on an Arduino prototyping shield. This simple solution worked remarkably well and was later introduced for the Nytol rig as well, in this case allowing switching between the filter and the rotary evaporator. The distinct disadvantage is that one valve only allows switching between two modules. Ultimately, a computer controllable manifold should be developed and implemented, allowing all modules that require temperature control to be serviced by one chiller.

A common laboratory operation that was initially disregarded is the drying of organic phases before evaporation. Usually a human chemist would use anhydrous magnesium sulfate or a similar desiccant to remove moisture in the form of both heterogeneous droplets and homogeneous solutions. During development of the Nytol synthesis, small amounts of water never posed a problem and would normally be removed by extended drying under vacuum on the rotary evaporator, so a drying module had not been implemented. However, during development of the Sildenafil synthesis it became apparent that there was an urgent need for this operation to be incorporated. It was reasoned that the simplest way of drying organic phases would be to pack a cartridge with desiccant and insert it into the flow path to the rotary evaporator. However, concerns were raised that having such a cartridge permanently

attached to the rotary evaporator may lead to unforeseen problems during transfer of material into and out of the distillation flask when drying is not required.

Thus, two six-way valves were used together to form a cartridge carousel (Figure 52). The central input of one valve was connected to the Backbone, while the central input of the other valve was connected to the rotary evaporator. The six output ports could then be connected to cartridges containing desiccant or be shunted using a length of tubing. This allowed interacting with the rotary evaporator as if it was connected by a simple tube as long as drying was not required. For drying operations, both valves would be switched to one of the cartridges, thereby inserting it into the flow path.

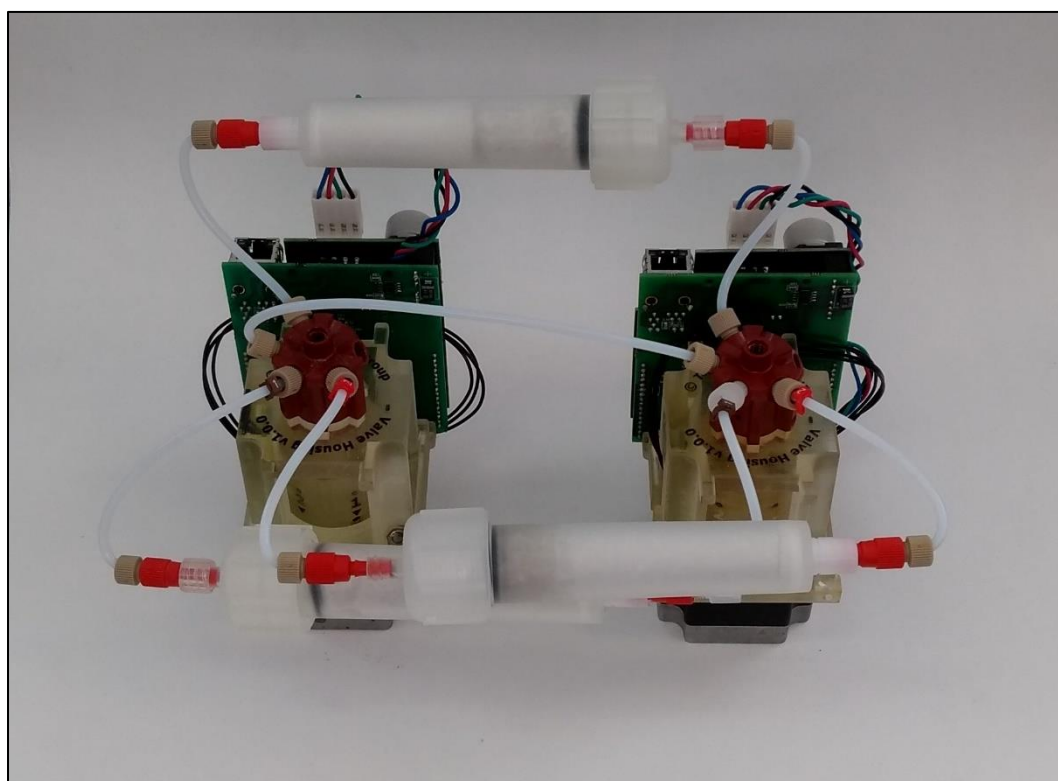


Figure 52: Photo of the cartridge carousel (left) and schematic of the connectivity (right).

Initially, magnesium sulfate and calcium chloride were investigated as desiccants, however, those salts tended to clump and block the cartridge. Therefore, molecular sieve (4 Å, 1-2 mm beads) was employed instead. The relatively large beads provided little flow resistance and would undergo no morphological changes during water uptake. Packing a 20 x 80 mm polypropylene column with freshly activated 1-2 mm

RESULTS AND DISCUSSION

diameter beads of molecular sieve was found to reliably dry every organic layer it was tested with. In the future, similar columns could also feasibly be used for activated charcoal or similar treatments.

An unexpected beneficial side effect of this cartridge carousel is that ports not used for cartridges could be used as additional chemical inputs. The code resulting from such a secondary use is unfortunately rather opaque. It involves switching the cartridge carousel to a position and then moving a volume from the rotary evaporator to a target vessel, which can be misleading if not commented properly. Also, it potentially interferes with the internal volume tracking. Thus, this approach should only be used in emergencies, if adding another Backbone unit is a substantial effort. However, as a quick-and-dirty workaround for method development is has been proven to be very useful.

To save space, the Backbone units were always placed on shelves at the back of the fume hood. Initially, those shelves were constructed from support rods and plastic sheets, which didn't make for a very professional and tidy appearance. As more platforms were built, and it became apparent that the Backbone architecture was not going to be changed in the near future, a custom shelf was designed and manufactured (Figure 53).

The shelf was dimensioned to fit the mounting studs already present in the fume hoods. The shelves consisted of two sheets of polypropylene stacked on top of each other. Polypropylene was chosen for its high chemical resilience. The lower sheet provided mechanical stability, while the upper sheet had cut-outs to fit either pumps or valves, thus keeping the units in place and neatly aligned. The shelf for the pumps was also fitted with a thinner plate in front separating the syringes from the pump body. This sheet served two purposes. First, it protected the pump in the case of a leakage or explosion of the syringe, which is not an entirely unprecedented scenario. Second, it kept the pumps from toppling over, as their centre of gravity was just about supported by their footprint.

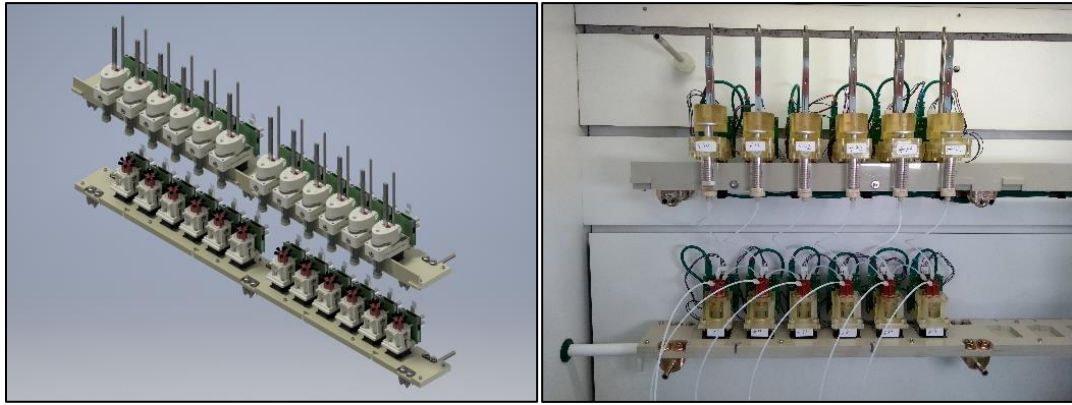


Figure 53: Custom shelf for pumps and valves. Left: Autodesk Inventor rendering. Right: photo of the shelf mounted in a fume hood.

The two sheets were joined by a number of bolts, forming a very rigid assembly. This assembly was then mounted to the studs using wire rope grips. Admittedly, this might not be the most efficient way of mounting the shelves, but it has the distinct advantage that these wire rope grips are inexpensive and readily available, therefore no complicated custom solution had to be designed and manufactured.

To make cable management easier, a slot was added for every device position, allowing the user to clip ethernet cables into the back of the shelf. A number of cable cradles was mounted to the bottom of the shelf, so the cables could be conveniently routed to the edge of the fume hood.

All polypropylene components were designed strictly two-dimensional, so they could be cut out from sheet stock on a waterjet cutter. Engineering drawings can be found in appendix II, , DXF files and a bill of materials are available from the Cronin Group upon request.

RESULTS AND DISCUSSION

7 SOFTWARE DEVELOPMENT

In parallel to the hardware development discussed in the previous chapter and the development of the chemistry discussed in the next chapter, the software controlling the devices as well as the entire platform had to be developed as well. This development effort broadly fell into three categories, which will be discussed in detail in the following subchapters. First, the firmware controlling the pumps and valves rewritten to improve reliability. Second, an interfacing strategy for the various third-party instruments had to be implemented. Third, an overarching control software had to be developed in order to execute chemical syntheses on the platform.

The firmware project was written largely in C, with some parts written in AVR Assembly code. The other two projects were written entirely in Python 3.

7.1 PUMP AND VALVE FIRMWARE

7.1.1 EXISTING HARDWARE SPECIFICATIONS AND FIRMWARE REQUIREMENTS

As was already mentioned in chapter 6.1 the pumps and valves were controlled by a custom PCB. This PCB had been designed earlier, and a major redesign was not possible due to time constraints, so it had to be used as-is. The board featured three major components: an Atmel ATxmega128A4U microcontroller (Figure 54), a WIZnet W5500 network chip, and a Trinamic TMC262 stepper motor controller. Circuit diagrams of the boards can be found in appendix III.

RESULTS AND DISCUSSION

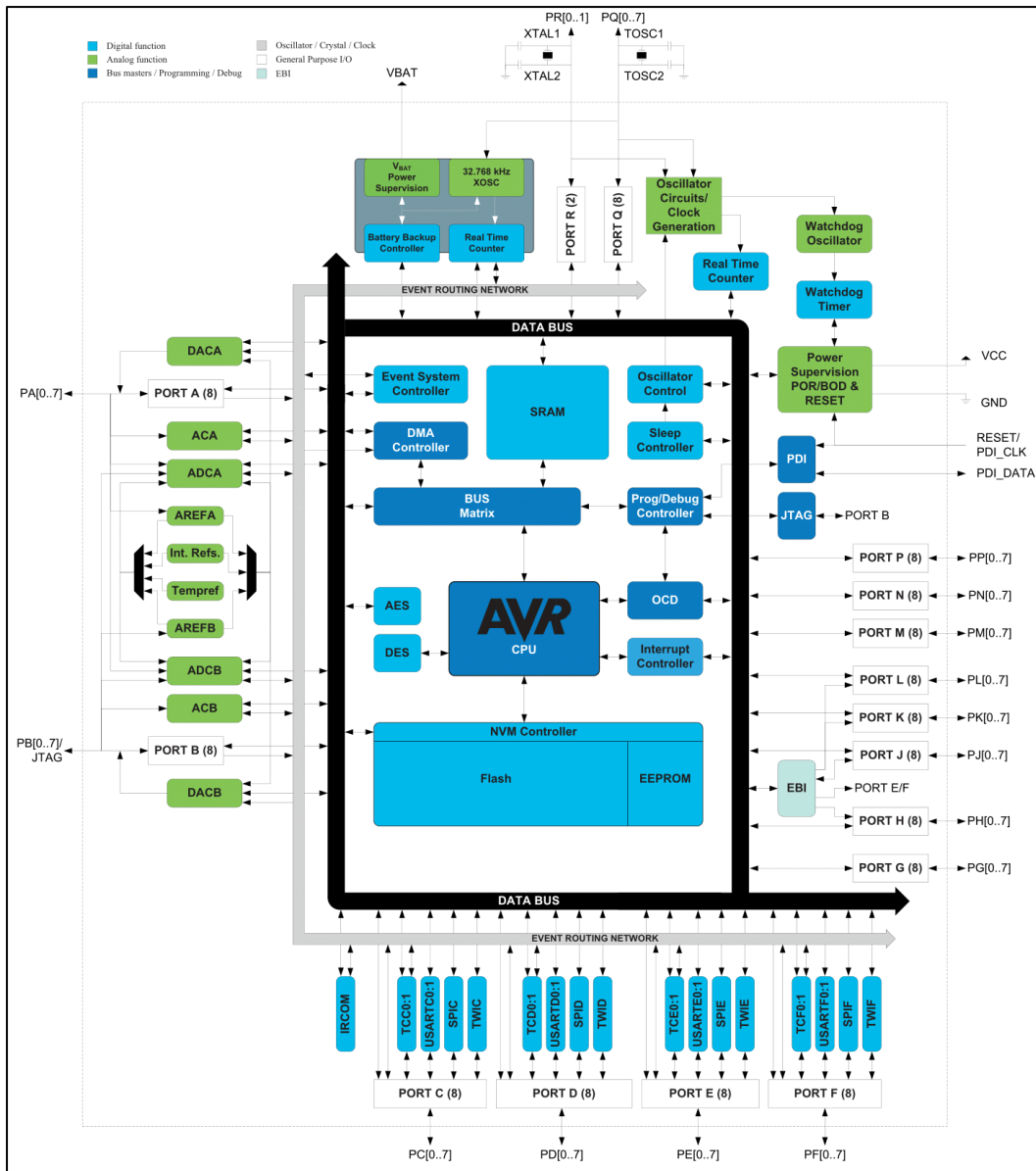


Figure 54: Block diagram of the Atmel® ATxmega128A4U microcontroller. Copyright held by Microchip Technology, Inc. Reproduced under the “fair use” exemption.

A firmware for the board had been written by Dr. Trevor Hinkley before I joined the project. This firmware was built on some peculiar choices, which will be introduced and discussed in the following sub-chapters. Overall, it was mostly operational, but had some bugs which severely hampered the usefulness of the devices. The most important issue was that, at random times, the devices would simply disconnect from the computer, and enter a state where only power cycling them would render them operational again. This was extremely inconvenient as it would regularly interrupt an

automated synthesis, with no way of recovering from the error except for manually restarting the devices and resuming the sequence. Other problems were encountered as well, but the disconnection issue was by far the most severe. Significant debugging effort was invested by several people, but ultimately to no avail. Thus, it was decided to abandon the old firmware and write an entirely new firmware from scratch.

7.1.2 GENERAL OPERATION AND REQUIREMENTS

On a top level, the requirements for the firmware were simple. The first functional block was communication with a controlling PC. The ethernet communication was largely handled by a dedicated network chip, the WIZnet W5500, which communicates with the MCU *via* SPI. Drivers for the chip were available from the manufacturer online, so the firmware merely needed to appropriately configure the chip on bootup, receive commands and send replies *via* SPI, and map incoming commands to their respective functions.

The second functional block was the control of the stepper motor. This was largely handled by a dedicated stepper driver chip, the Trinamic TMC262. Again, drivers were available from the manufacturer online. The chip required a sophisticated setup in order to optimally control a specific model of stepper motor, yet once this configuration was done, the interaction was limited to sending signals to three pins: Enable, STEP and DIR. The Enable pin would power the motor when pulled to low and cut off power when pulled high. The DIR pin controlled the direction of the rotation. The absolute direction of rotation depended on the wiring of the motor, but the pin state of DIR would toggle between the two directions. The STEP pin would advance the motor by one full, half or microstep, depending on the current configuration, on every rising edge detected at the pin.

The third and most important functional block was the control of device specific movements. This means for the pump, moving the plunger by a given volume, and for the valve, finding and moving to a given valve port. This block included algorithms

for movement and position finding, counting motor steps, and reading the Hall effect sensor to determine valve ports or the home position of the pump.

Based on this assessment, it was decided that a real-time operating system was not required and would only represent an unwarranted complication. Most operation could be offloaded to the peripherals, which meant that a simple interrupt-driven design would be sufficient.

7.1.3 MCU INITIALISATION AND GENERAL SETUP

Before taking on any of those functional blocks, the microcontroller required some basic setup itself. After bootup, the system clock had to be initialised, and all GPIO pins had to be configured. The SPI interface used for communication with the network and motor driver chip needed to be initialised, as did the UART interface used for serial communication with the PC for debugging.

In the old firmware, all those tasks were accomplished by custom code writing the required bit masks to the appropriate registers. While this is, in principle, a perfectly valid approach, it leaves some room for errors. More importantly, however, it is unnecessary since the chip manufacturer Atmel (now Microchip Technology Inc.) provides drivers for its microcontrollers.

Atmel offers a specialised Integrated Development Environment (IDE) named Atmel Studio.¹²⁵ It is based on Microsoft's Visual Studio, it contains a readily set up compiler toolchain for AVR and SAM architectures, and a vast range of libraries for all chip families. It supports C/C++ and assembly code. A cloud-based code repository, the Atmel Software Framework or ASF allows the user to download and include libraries for many basic tasks like sending and receiving data through SPI or UART, using peripherals such as counters or the ADC, or controlling GPIO pins.

Using those ASF functions, the initialisation after bootup commences with initialising the Programmable Multilevel Interrupt Controller (PMIC) and disabling all interrupts until the setup is complete. Then, the system clock is initialised, followed by the GPIO ports.

Next, the motor controller board is enabled after a 500 millisecond delay. As mentioned in chapter 6.1.1 the control board consists of two independent boards, a mother board featuring the MCU and network chip, and a motor driver board featuring the stepper controller and the associated MOSFET stage. As driving the stepper motor requires controlling the current through “chopping”, which essentially means switching it on and off really fast, a considerable level of high frequency noise is generated by the motor driver board. As the power is supplied through the same physical wires as the data, this noise would severely impair data communication. Therefore, an LC low-pass filter containing four capacitors of 220 μ F each is implemented on the motor controller board. The distinct disadvantage of this design is that this large capacitance requires charging upon power-on, which results in a large inrush current, which may overload the power supply, especially if multiple devices are powered on simultaneously.

To counteract this behaviour, a normally closed power MOSFET is inserted into the circuit. The transistor is shorted by a 680 Ω resistor which acts as a current limiter and allows the capacitors to charge slowly. After MCU bootup and an additional 500 milliseconds, the MOSFET would open and allow the motor controller board to draw maximum current. Figure 55 shows a typical charging process by plotting the voltage across the capacitor (which is proportional to the charge of the capacitor according to Equation 1) against the time. The graph shows slow charging at first, followed by a steep increase once the MOSFET is opened.

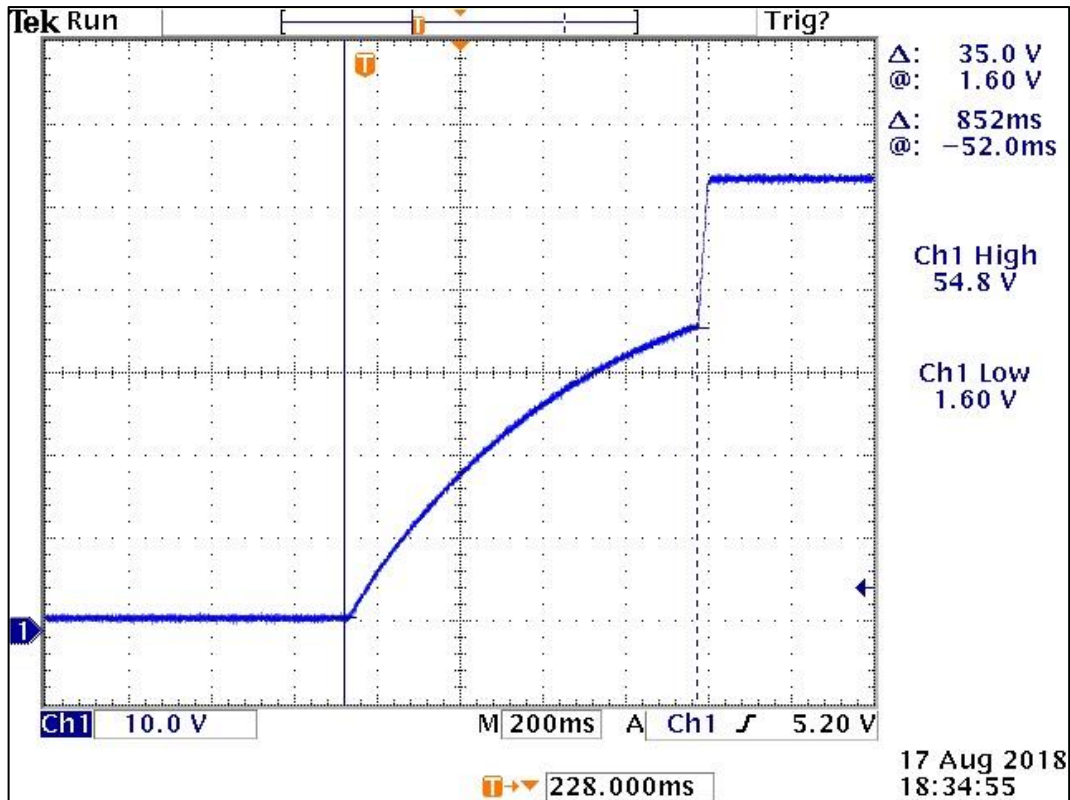


Figure 55: Oscilloscope measurement of the charging of the capacitors by measuring the voltage across the capacitor. X axis increments are 200 ms per square, Y axis increments are 10 V per square.

After enabling the motor controller board, all GPIO pins are configured and set to their initial values. Then the UART and the SPI connection are initialised. This is followed by configuring the PWM counters used for stepping, as well as configuring the ADC. Those two configuration steps will be discussed in detail in chapter 7.1.6. Subsequently, the device configuration is read from the EEPROM (see chapter 7.1.9). While doing so, the motor profile is read from the EEPROM as well and sent to the TMC262. Stepper driver configuration and motor profiles will be discussed in detail in chapter 7.1.6. This is followed by initialising the network configuration which will be discussed in detail in chapter 7.1.4.

As last setup action, the watchdog timer is initialised to 4 seconds. The watchdog timer continually counts down from its defined interval, and when it reaches zero, it resets the chip. Some regular event, in this case a UDP broadcast coming from the PC, has to reset the timer before it runs out. The purpose of this timer is to stop the

device operation should the connection to the PC be lost, or the control software hang. At this point, however, the Watchdog is not yet started, as this would otherwise result in a four second boot loop as long as the device is not connected to a PC, or the PC is not yet attempting to communicate.

After successfully completing the entire initialisation, interrupts are globally enabled, and the firmware enters a loop where it waits for commands received either through the UART or the ethernet connection.

7.1.4 ETHERNET COMMUNICATION

On a physical level, the ethernet communication is handled by a dedicated chip, the WIZnet W5500. In a nutshell, it handles the low-level aspects of the TCP/IP implementation as a black box and offers a high-level interface which enables the user to send and receive messages through TCP, UDP, or a number of other communication protocols. It offers eight sockets, so a maximum of eight distinct connections can be maintained simultaneously, and 32 Kbyte buffer memory, which is shared among all sockets. Communication with the MCU is handled *via* SPI.

The previous firmware used TCP to receive commands from the PC, and UDP to receive a “keepalive” broadcast to regularly reset the Watchdog timer. TCP, or Transmission Control Protocol, is a reliable peer-to-peer communication protocol. It features a range of message verification tools and redundancies which make it resilient against data loss or corruption, at the expense of speed. This makes it ideal to transmit commands which must be delivered faithfully. UDP, or User Data Protocol, however, is a faster protocol lacking those safeguards, which allows peer-to-peer connections as well as broadcasts (i.e. one sender, multiple recipients). This makes it ideal for transmitting Watchdog resets. A PC can just broadcast those keepalive messages into a subnet without necessarily having to worry about connecting to each device individually, and as long as those broadcasts happen frequently enough, a device can cope with the occasional lost or corrupted message.

So far, this made sense. However, the old firmware layered an additional communication protocol on top of the TCP. This layer, Open Sound Control or OSC,

is a communication protocol originally conceived for the communication with musical instruments and sound synthesisers.¹²⁶ It remains unclear to the author why this particular protocol was chosen, or for that matter why another layer of complication was introduced at all. Furthermore, the implementation of the OSC standard used in the old firmware project seems to have been written by my predecessor from scratch, so there might well be errors hidden in the (largely undocumented) code.

Another peculiar choice regarding the network communication was the use of DHCP. This Dynamic Host Control Protocol dynamically assigns IP addresses to devices in a network. DHCP is a standard tool routinely used to manage computer networks, and in and of itself it is not unusual. What makes it unusual with regards to the pumps and valves is that it seems entirely unnecessary. Every device requires a unique identifier in order to be addressable by the control PC, and a fixed individual IP address appears to be a straightforward solution. The old firmware instead featured individual device IDs and dynamically assigned IP addresses. DHCP is advantageous in large networks where manual IP management becomes unwieldy, or for mobile devices such as laptops or phones which regularly travel between multiple different networks. Either case appears unlikely for the pumps and valves, so there is no readily apparent advantage to using DHCP. The distinct disadvantage, however, is that it adds considerable overhead to the firmware. The ATxmega family is one of the most powerful 8 bit microcontrollers on the market, but it is still a constrained system with limited resources, so additional loads should only be added after careful consideration. Furthermore, it again appears as if the DHCP implementation in the old firmware was written by my predecessor rather than taken from a tested library, so it is entirely conceivable that the disconnection problems encountered earlier are related to the use of DHCP.

Based on this assessment it was decided to retain the use of TCP for commands and UDP for keepalive broadcasts, but without an additional intermediate layer. It was also decided to assign static IPs to every device and implement a mechanism to change the IP at runtime without having to reprogram the MCU.

WIZnet provides a driver package on their website, which offers high-level interfaces for all important functions. In order to initialise the W5500, several tasks have to be completed. First, several functions related to SPI communication have to be registered with the driver library. The user passes pointers to the functions in question to the library, which in turn uses those pointers to perform communication tasks in a chip-agnostic way.

Then, the 32 Kbyte buffer memory has to be partitioned between the sockets. For reasons of simplicity, the memory was distributed evenly, resulting in 2 Kbyte of buffer for both transmit and receive on every socket. Since ultimately only two sockets are in use, the available memory could in principle be allocated in full to them. However, it was reasoned that first of all 2 Kbyte ought to be enough for anybody, given that the intended data consists of merely a handful of ASCII characters per command, and second of all, leaving space for future expansion is a prudent approach.

Next, the network configuration is read from the EEPROM and sent to the W5500. The network configuration consists of the IP and MAC addresses as well as a few other less important parameters, which all have to be loaded into their respective registers on bootup. This data is stored in the EEPROM as opposed to the program memory in order to enable the user to safely change the addresses during runtime. A default value is loaded into the EEPROM when the MCU is programmed, so the device is immediately network enabled. After programming the device, the user is encouraged to assign a unique IP using the API (see chapter 7.1.10).

Next, the interrupts have to be configured. The W5500 features one pin which is pulled to low if any of a number of predefined events happen. This pin is connected to an input pin of the MCU, which generates an interrupt if the pin is low, thereby prompting the MCU to deal with the network event. The W5500 uses a system of bit masks to determine which events result in a low pin state. One bit mask determines which of the eight sockets may trigger an interrupt, while one bitmask per socket determines the type of event that is permitted to trigger an interrupt. If both bits are set for any given event, and the event occurs, a bit is set in the global interrupt register, and the pin is pulled to low. It falls to the MCU to clear the bit after servicing

the interrupt, and the pin will remain low until all bits in the interrupt register are cleared. The Interrupt Service Routine associated with the W5500 will be described in detail a bit later. For this firmware, the interrupts associated with two sockets were enabled, one for the TCP communication, and one for the UDP broadcast. For the TCP socket, connection of a new client, disconnection, receiving a message, and socket timeout were enabled to send interrupts. For the UDP socket, merely the receiving of a new message was enabled.

As last step, the two sockets were configured for TCP and UDP, respectively, and assigned a port number. Then, the TCP port was set into listening mode. In this mode, the device would act as a passive server, allowing exactly one PC to connect to it as a client and issue commands.

Having been set up this way, the W5500 will now listen for incoming TCP connections or UDP messages. Once a TCP connection is established, or a UDP message is received, it will trigger an interrupt for the MCU. The ATxmega family features a Programmable Multilevel Interrupt Controller or PMIC, which allows the user to assign one of three priorities to an interrupt source, low, medium, and high. Any Interrupt Service Routine (ISR) can be interrupted by sources with higher priorities, while interrupts with equal or lower priority are logged by the PMIC and are serviced once the current ISR returns. The W5500 interrupt is assigned medium priority. The reasoning here is that interrupts for the movement routines are more time critical and must be serviced at the earliest convenience. Since the W5500 drivers globally disable interrupts during critical transactions, and the movement routines share no memory with the network communication, the W5500 ISR can be safely interrupted.

Said ISR, when executed, will first request a copy of the interrupt register of the W5500 in order to determine which event caused the interrupt. To that end, it will then go through each enabled case and check if the corresponding bit is set, and if so, service the event and clear the bit. This will continue iteratively until all bits are cleared. This is to avoid a deadlock if another event occurs while one interrupt is serviced.

The first possible event is a new TCP connection. In that case, a debug message containing the client IP is sent through the UART connection. Then, the Watchdog timer is enabled. The reason is that as soon as a TCP connection is established, a move command could feasibly be issued, so the safety mechanism has to be armed by that point. As a visual cue to signify a successful connection, an LED on the board is illuminated.

The next case is if data is received *via* TCP. This means a command has been issued and needs to be dealt with. After verifying that data is available and that it fits into the allocated buffer on the MCU, the command is read from the W5500 and passed to the command parser (see chapter 7.1.5).

The next case is a client disconnecting from the device. In this case, the LED is switched off and a debug message is logged to UART. If the firmware is compiled in debug mode, the Watchdog timer is disabled to aid development and debugging. The release version, however, would leave the Watchdog timer enabled, to ensure safe operation throughout. The TCP socket is then returned to listening mode, which means as long as the UDP keepalive is sustained, another client could connect and continue operating the device.

The last case for the TCP socket is a timeout, in which case a debug message is logged. Since the Watchdog timer ensures safe operation in case the client hangs, this was deemed sufficient.

For the UDP socket, only a new message is enabled to issue interrupts. In this case, after checking if data is available and if it fits into the buffer on the MCU, the data is read from the W5500 and compared against the Watchdog reset keyword `reset_wdt`. If it compares positively, the Watchdog timer is reset. Otherwise, nothing happens. The incoming data will only be compared up to the length of the reset keyword to conserve CPU time. While a synthesis platform can reasonably be expected to be a dedicated, closed network, in modern computers it should be assumed that other background programs might send messages into any available network. This may be counteracted by locking the network down using firewall settings, but the easier path is to design the firmware in a way that simply ignores

any and all transmissions that are not the expected keyword. Yet, there is still the remote danger of consuming all CPU time if UDP transmissions are received faster than they can be serviced. The time it takes to clear a UDP keepalive interrupt was measured at 2.7 ms, at a default frequency of one keepalive message every half second, this utilises around 0.5% of all CPU time. Thus, it seems unlikely an interfering application could overwhelm the device, and even if it would, the Watchdog timer would run out and safely stop operation.

7.1.5 THE COMMAND MAPPER AND THE FORMATTED NETWORK PRINT UTILITY

It was decided to use human-readable strings as commands as that would make it easy to debug misbehaviour, and also enable people to conveniently implement APIs for interfacing with the devices. In the course of this work, an API in Python 3 was developed, but should anyone require device control in a different language, a simple, string-based protocol makes this as straightforward as possible.

The difficulty on the microcontroller side is to implement an efficient function that parses a string containing a variable-length command and a variable number of arguments of varying types and executes the associated routines. After some research it was decided to adopt an approach published by Mark McCurry.¹²⁷

The problem of the varying argument types was solved by defining a union of all possible types, called `arg_t`. That would be: signed and unsigned integers of 8, 16, 32, and 64 bits of length; floating point numbers; and strings (char pointers). Those types were assigned single letters according to the following scheme: c, i, l, e stand for char (8 bit integer), int (16 bit integer), long (32 bit integers) and extra long (64 bit integer), f for float (floating point numbers) and s for string. For the integers, lowercase letters denote signed types, uppercase letters denote unsigned types.

All routines which should be exposed to the user were cast into functions of a uniform structure, which take an `arg_t` pointer as argument and return `void`. This might be as simple as defining appropriate functions that return a call to an internal function which returns `void`. In other cases, like for writing device configurations, the arguments had to be unpacked and passed to a function explicitly. Either way,

those uniform functions were then arranged into a dispatch table. This table consists of structures containing the following pieces of information: a command keyword in the form of a string, a pointer to the associated function, a string defining the number and type of the associated arguments encoded in the single letter code described above, and a short docstring explaining the command and associated routine.

An incoming command consisting of a keyword and a number of arguments separated by a delimiter (by default a space character) is tokenised into substrings by splitting it at the delimiters. The first substring containing the keyword is matched against the keywords in the dispatch table. If a match is found, the rest of the tokens are parsed into their appropriate data types using the argument string obtained from the dispatch table and the results are stored in a list of `arg_t` unions. If the number of arguments provided does not match the expected number of arguments, or if an argument cannot be converted according to the required type, an error is raised. If the arguments parse without error, the function obtained from the dispatch table is called with the parsed arguments. All string-related operations such as tokenising or matching was accomplished using the `<string.h>` standard library.

Being able to parse and execute incoming commands, a simple way of sending replies was required. To that end, a function was written which allows the user to send formatted strings *via* TCP similar to how the `printf()` function of the `<stdio.h>` library sends formatted strings via a serial interface. This utility, named `netprintf()`, uses the `vsnprintf()` function of the `<stdio.h>` library to compile a string with format specifiers and a list of arguments into a formatted string, which is subsequently sent to the client *via* TCP using the W5500 driver library. This function was used extensively across the firmware wherever replies *via* the ethernet connection were required.

7.1.6 STEPPER MOTOR CONTROL

Both the pump and the valve use bipolar stepper motors for movement and positioning. Those motors are controlled by a dedicated chip, the Trinamic TMC262. Before going into the details of this chip, let us briefly discuss the operating principle of stepper motors. Unlike other electric motors, stepper motors do not revolve continuously, but instead advance in a stepwise fashion in one direction or the other, hence the name. The step angle is known, so in order to move a motor by a given amount, the user merely needs to advance it by a given number of steps. Figure 56 shows a simplified model of a bipolar stepper motor. A bar magnet is mounted on a shaft (small circle in the middle) and surrounded by four electromagnets. The windings of the electromagnets A and A1 as well as B and B1, respectively, are connected in series in such a way that the directions of their magnetic fields align. Consider energising the coils of A and A1. The bar magnet will line up as shown in the figure. Now consider de-energising A and A1, and energising B and B1. The bar magnet will rotate by 90° clockwise or counter-clockwise depending on the direction of the current and hence the magnetic field and align itself with the B coils. De-energising B and B1, and energising A and A1 again, but this time with the current and hence the magnetic field in the opposite direction, will cause the bar magnet to continue its rotation by another 90°. Thus, the shaft can be rotated in 90° increments.

Now consider instead of de-energising one pair of windings when energising the other, both pairs are energised at the same time. The bar magnet will orient itself between the windings at a 45° angle. When one winding is de-energised, the bar magnet will align itself fully with the other winding, thus travelling by 45°. Re-energising the first winding again in the opposite direction will cause the bar magnet to align itself between the windings again, travelling by another 45°, and so forth.

So far, all cases assume the same current in all windings. However, by applying different currents to the two winding pairs, the bar magnet can be made to align itself closer to one pair or the other, with the angle being proportional to the current ratio. This way, rather than advancing 90° or 45° at once, the shaft can be moved smoothly to any position. This mode of operation is called microstepping. Practically,

the current is not tuned continuously, but rather increased or decreased in small increments, resulting in a number of microsteps between the full steps.

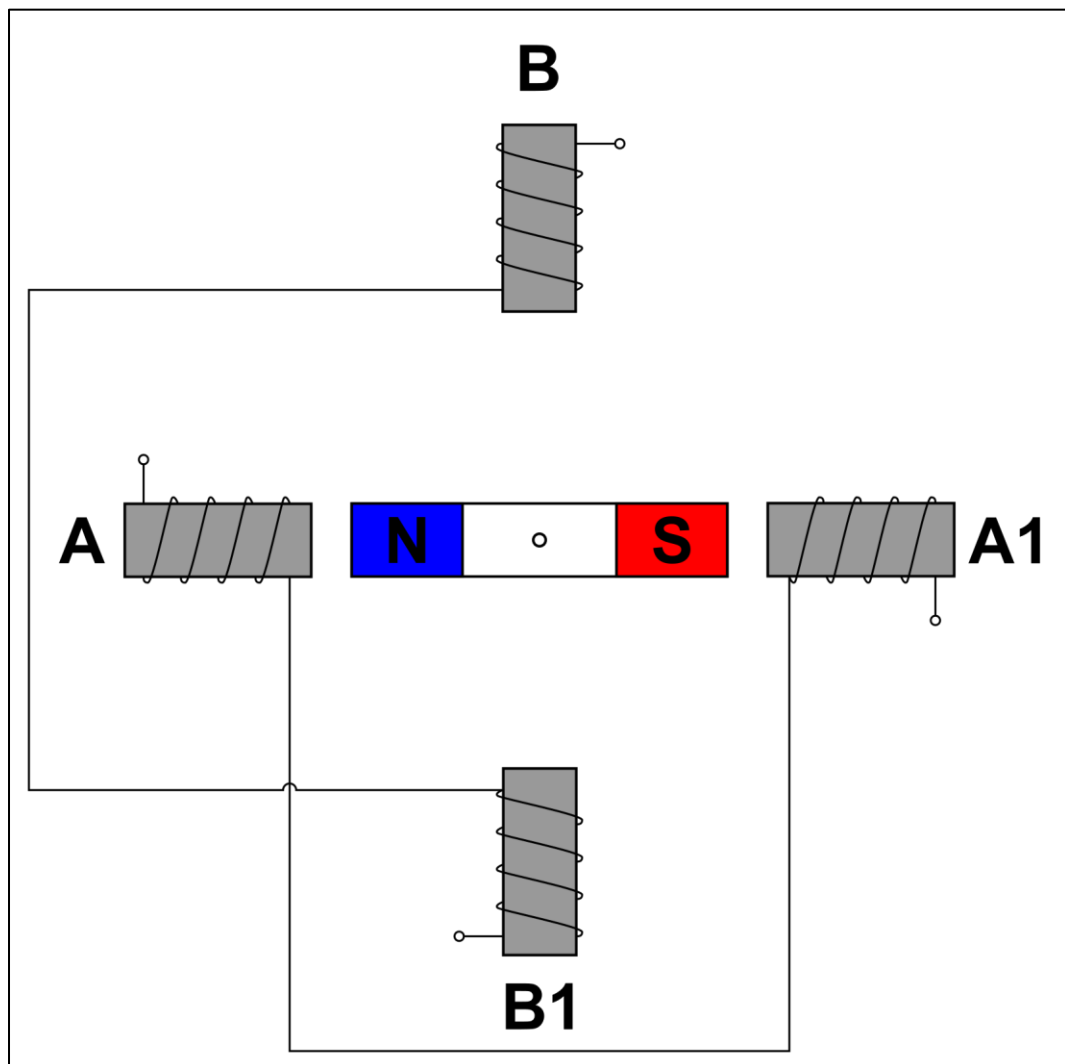


Figure 56: Simplified model of a bipolar stepper motor.

The configuration shown in Figure 56 is merely a model to explain the principle. Real stepper motors are built slightly differently, but the general mode of advancing it through the steps remains the same. Real stepper motors have step angles significantly smaller than 90° , a commonly encountered value is 1.8° or 200 steps per revolution. The number of microsteps is usually a power of two, with 256 microsteps per full step being a commonly used resolution. Practically, the maximum resolution of a stepper motor depends on many factors, and microstepping has advantages and disadvantages compared to full stepping. A designer has to weigh those against each

other and decide on an operating mode, and in the case of microstepping, on a sensible microstep resolution. In this work, with more zeal than judgement, a microstepping resolution of 256 microsteps per full step was chosen.

In order to tune the coil current when only a constant voltage is supplied, the controller chip switches the power on and off in fast succession and makes use of the fact that a coil, being an inductor, can store energy for a short time. When voltage is applied the current in the coil rises, until it reaches the target current. Then, the voltage is switched off, and because the coils have an inductance, the current will fall only slowly. When it crosses a defined lower threshold, the energy stored in the coil is “topped up” by briefly applying voltage again. This way, the average current in a coil can be held near the desired value. The TMC262 uses eight external MOSFETs for that task, which form two H bridges, one per coil. An H bridge is a circuit of four switches which allows a source voltage to be applied to a load in either direction. In fact, the TMC262 uses an even more involved switching cycle than described above to optimally shape the current, but those details reach beyond the scope of this work. The interested reader is referred to the datasheet of the chip. Practically, the chip handles microstepping as a black box, and the user only needs to supply pulses to one of the TMC262’s pins in order to advance the motor.

In addition to microstepping and current control, the TMC262 also offers features related to load measurement. By measuring how the coil current reacts to the voltage switching cycle, the chip is able to deduce the motor load. Based on that, in theory it can dynamically adjust the current to compensate for increased load, and thereby decrease the overall power consumption and associated heating for low loads. It can also, in theory, report a stall, that is when the maximum load is exceeded, and the motor gets stuck. Practically, in this particular application the move speeds were generally too slow to allow for reliable operation of those features. Thus, the dynamic power scaling was turned off entirely, and the stall detection was only used for hard homing the pump (see chapter 7.1.7).

In order to perform all those functions, the TMC262 required a number of settings. A full list of all parameters is available in the chip’s datasheet. Some parameters like the microstep resolution are user defined. Settings related to optimally shaping the

coil current depend on the characteristics of the motor and can be approximated using calculator tools provided by Trinamic. Fine tuning could be performed by monitoring the coil current with an oscilloscope, but for the purpose of this work it was found that the default values provided by the calculator worked satisfactorily. The stall detection required experimental tuning. In the firmware, all parameters were stored in a structure called a motor profile. Upon bootup, those settings were sent to the TMC262 *via* SPI.

Once this setup is complete, driving the motor requires three pins as described in chapter 7.1.2. The Enable pin is used to switch the power to the motor on or off. During operation, Enable must be pulled low to switch on the power. When the motor is idle, Enable is pulled high to switch off the current and reduce energy consumption, as well as avoid excessive heating. The DIR pin determines the relative rotation direction. The absolute direction, i.e. clockwise or counter-clockwise, is also determined by the polarity of the motor cables, so the device configuration (see chapter 7.1.9) holds a variable denoting the positive direction. If a device is found to work the wrong way, i.e. a pump aspirates when it should dispense, or a valve rotates through the positions in the wrong order, the user can change the variable in the configuration and the DIR pin polarity is inverted.

The STEP pin senses digital pulses and causes the TMC262 to advance the motor by one full/half/microstep, depending on the settings, every time an edge is detected. The user can choose whether only rising edges or rising and falling edges are considered. Since this application only requires low stepping frequencies, the firmware uses rising edges only.

In order to generate those edges, one of the ATxmega's timer/counters was used to generate a Pulse Width Modulation (PWM) signal. The frequency of this signal is equal to the desired stepping frequency, and the duty cycle is arbitrarily set to 50%. Those timer/counters are peripheral devices on the microcontroller which count pulses of an input waveform. The input waveform can be a clock signal, or any arbitrary signal generated by other components on the PCB. A user may set a maximum value and up to four compare values, and the timer/counter will generate an interrupt or an event once those values are reached. Generating PWM signals is a

standard application of timer/counters, and the ATxmega provides a dedicated mode for it. The main advantage of using timer/counters for this task is that they operate entirely independent of the CPU. Once the required parameters are loaded into the corresponding registers, the peripheral will continue to produce a PWM waveform until told otherwise, without any further involvement of the CPU. This conserves processor time and frees up resources for other tasks.

In order to position the motor accurately, those pulses have to be counted. To do so, another timer/counter was used in conjunction with the event system. Events are a feature of the ATxmega family used for signalling between different peripherals. In this case, the timer/counter producing the PWM would signal a second timer/counter every time a rising edge is emitted and thus a step is made, and the second timer/counter would count up by one step. The timer/counters of the ATxmega128A4U can hold up to 16 bit of information or 65,536 steps. At a resolution of 256 microsteps per step, and a step angle of 1.8° , this works out to 460.8° of movement or roughly one and a quarter revolution. This is adequate for the valve, but not for the pump. The solution recommended by Atmel for this problem is to concatenate two 16 bit timer/counters. This is done by leveraging the event system much in the same way as for counting pulses. Once the “low” counter overflows, the “high” counter is incremented, thus effectively creating one 32 bit timer/counter. This allows for a total of 2^{32} or 4,294,967,296 steps or not quite 84,000 revolutions. For reference, the lead screw used in the pump has a pitch of 2 mm, so this would equate to almost 168 m of travel.

Using this setup, an algorithm for moving the motor by a defined number of steps can be devised as follows. During bootup, the PWM timer/counter was set into waveform generation mode and configured to output the waveform to the STEP pin. The low side of the step counter was set up to count rising edges as described above, and the high side was configured to count low side overflows. To initiate the move, the PWM timer/counter was set to the desired stepping frequency and 50% duty cycle. If fewer than 65,536 steps were required, the low side step counter was zeroed, the maximum value was set to the required number of steps, and the overflow interrupt was enabled. Then, the Enable pin was pulled to low, the DIR pin

was set as required, and the PWM was started. From that point on, the peripherals would step the motor without CPU involvement. Once the last required step was made and the step counter overflowed, an interrupt would be generated to signal that the move is done. The associated ISR would contain the device-specific positioning algorithm, which will be discussed in chapters 7.1.7 and 7.1.8, and the CPU would issue the next command as required.

If more than 65,535 steps were required, the 32 bit counter had to be used. Here, things were a bit more complicated as both counters had to reach a target value. To understand the algorithm, consider Figure 57. It shows the number 500,000 encoded in binary, separated into two 16 bit chunks, representing the high side counter and the low side counter. Note how the high side counter alone reads binary 111 while the low side counter alone reads 1010 0001 0010 0000.

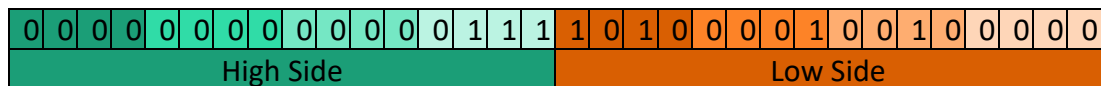


Figure 57: The number 500,000 encoded in binary. “High Side” and “Low Side” refer to the leftmost 16 bit and the rightmost 16 bit, respectively.

Both counters start at zero. The maximum value of the low side counter is set to $2^{16}-1$ or binary 1111 1111 1111 1111. Every time it reaches this value and tries to increment again, it overflows and starts at zero again. At the same time, an event is generated which increments the high side counter. This procedure is repeated until the high side counter reaches binary 111. Then, the low side counter counts until it reaches 1010 0001 0010 0000. Then, an interrupt should be generated to signal the successful completion of the 500,000 steps.

To that end, the ATxmega’s timer/counters allow the user to set a compare value. If the current counter value matches this compare value, an interrupt is generated. Thus, the low side counter compare value was set to 1010 0001 0010 0000. However, on its own, this compare match would trigger an interrupt every time the counter

reaches this value, and not just once the high side counter had reached 111. Thus, the compare value and the corresponding interrupt were set up during the initialisation of a move, but not activated. The high side of the counter was then configured to generate an interrupt once it reached 111, and the corresponding ISR written in Assembly would activate the interrupt on the low side counter. Thus, during operation, the low side counter would run from zero to maximum until the high side counter reached 111, then the compare interrupt would be enabled, the low side counter would continue to run until it reached 1010 0001 0010 0000 and then create an interrupt, precisely after 500,000 steps.

Practically, this algorithm worked very well. Tests with a logic analyser capturing the generated pulses showed that the requested number of steps was generated faithfully, and the ISR switching on the compare match interrupt was very fast. The ISR itself clocked in at less than half a microsecond, and the overall time between the rising edge and the end of the ISR was measured at around 2.7 μ s, which is fast enough to not cause any delays or step losses during normal operation.

When operating a stepper motor, it is advisable to gently ramp up the speed at the beginning of the move and ramp it back down at the end. This is to overcome the inertia of the rotor, and any inertia or friction in the system. Without gentle acceleration, the motor can lose steps before it starts moving, thus decreasing the positioning accuracy, or if the step change is too drastic, even fail to move at all and instead just oscillate in place. A similar principle applies to the deceleration, where the inertia of the moving system has to be absorbed. An energised stepper motor can resist turning only up to a certain limit, the so-called holding torque. If the inertia of the moving system is greater than the holding torque, the motor is jerked past its current position and the system moves one or more full steps past the theoretical step count. Those inaccuracies can mount up and eventually lead to significant positioning problems.

The old firmware used a complicated, adaptive, non-linear acceleration and deceleration curve which was calculated in real time. This curve required advanced mathematics which put unreasonable load on the CPU. The acceleration and deceleration phases are particularly time critical tasks, so the CPU load should be

kept to a minimum. One practical problem which was encountered with pumps running the old firmware was that the device would sometimes hang at the beginning or the end of a step, leading to the Watchdog eventually resetting the device. While never unequivocally proven, the most plausible working hypothesis was that the calculation of the acceleration curve took so much time that interrupts kept piling up until the backlog got longer than the Watchdog interval and the device was reset.

Thus, a simple linear acceleration and deceleration curve (Figure 58) was implemented and all required calculations were performed before commencing the move. Loading values from memory is considerably faster than performing divisions, so pre-calculating the acceleration and deceleration profile and storing it in memory is a much more efficient approach.

To that end, a safe starting frequency, a fixed number of acceleration steps, and a fixed acceleration resolution (W in Figure 58) were stored in the device configuration (see chapter 7.1.9). The acceleration steps describe the overall number of motor microsteps over which the motor should accelerate, while the acceleration resolution specifies the number of microsteps per plateau. Thus, those two numbers together specify a number of plateaus over which the motor is accelerated. Using a fixed number of speed plateaus over which the motor is accelerated and decelerated results in a variable acceleration rate depending on the travel speed, but it ensures short acceleration and deceleration phases. A fixed acceleration rate may perform better in some regards, but it would also mean moves at high speeds have long phases of lower and generally poorly defined speeds at the beginning and end which was seen as undesirable for the application in question.

When a move over a given number of steps at a certain speed was requested, the firmware would calculate the number of plateaus during the acceleration and deceleration from the acceleration steps and resolution. It would then calculate the speed increment (H in Figure 58) from the starting speed and the travel speed and store the speed increment and number of plateaus in memory.

The movement routine would then set the PWM timer/counter to the starting frequency and the step counter to the acceleration resolution W and start the move.

Once W steps were made, the step counter ISR would increment the speed by the stored value, decrement the number of remaining plateaus by one, and start the PWM again for another W steps, and so forth. Once the remaining speed increments reached zero, the ISR would instead set the speed to the requested travel speed and the step counter to the (pre-calculated) number of travel steps. At the end of the free travel, the acceleration procedure would be repeated in the opposite direction to decelerate the motor below the safe speed, before the move was concluded. The exact operation of the step counter ISR differed a bit between the pump and the valve and will be discussed in detail in the following chapters 7.1.7 and 7.1.8.

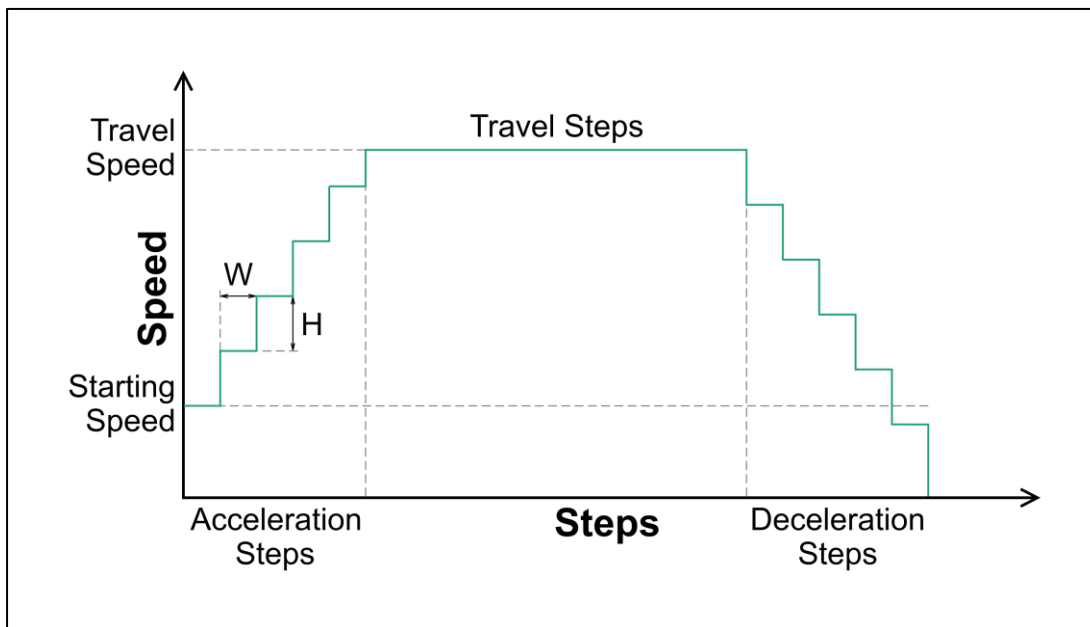


Figure 58: Diagram showing the acceleration, travel and deceleration profile for the stepper motor. W = Acceleration resolution. H = Speed increment. “Steps” refers to the motor microsteps.

The performance of the algorithm was tested by toggling a GPIO pin at the beginning and end of the step counter ISR and capturing the pin state as well as the generated PWM waveform with a logic analyser. The tests showed that the ISR execution time was short enough to not cause excessive delays between the plateaus (generally in the 60-80 μs range), and that the number of rising edges generated was precisely the

number of microsteps requested. The capture files in question are available from the Cronin Group upon request.

7.1.7 PUMP POSITIONING ALGORITHMS

For the pump, three movements were required: move absolute, move relative, and move home. A fourth movement, hard homing, was included later. Absolute movement should position the pump at a given volume between zero and the maximum syringe volume. In contrast, relative movement should move the pump by a given volume up or down, as long as the command would not move it below zero or above the maximum volume. While in practical applications this mode is rarely used, it was decided to implement it nonetheless as the extra effort was negligible. Move to home should prompt the pump to move towards zero without counting steps, and to stop when the Hall effect sensor detected the home position. Hard homing would work much the same way, except that it should stall the pump against the base and use the TMC262's stall detection to establish a true zero position. It should then back away from this hard stop by a predefined number of steps and record the resulting position as new home position. This algorithm should counteract sensor drift or shifts in reading due to different magnetic environments.

The old firmware used a globally set move speed for all movements, and changing that speed involved writing to the EEPROM. This meant sending a separate command every time a different flow rate was required, which was clearly inconvenient. Thus, it was decided that all movements should take the movement speed as parameter. Furthermore, the old firmware handled move speeds strictly in steps per seconds. Experimentalists however are usually interested in achieving a flow rate in millilitres per minute, or in order to avoid floating point numbers, microlitres per minute. For the old firmware the conversion from mL/min to steps per second had to be done by the calling software which was inconvenient and error-prone, so for the new firmware the conversion should instead be handled by the device itself. The added benefit of this approach was that the data required for the conversion, such as step angle, lead screw pitch, and syringe size would be contained in the device configuration, and therefore travel with the device.

The absolute movement routine would first check if the device had any errors (see chapter 7.1.9) and if the device was in the quiescent state. The latter was to avoid interrupting an ongoing move. Then it would check if the supplied position in μL was zero, in which case a call to `move_home()` would be returned. Otherwise, the target volume was converted to a position in steps and the result was checked against the maximum position of the syringe. If the move would exceed the maximum position an error would be raised. Otherwise, the current position and the target position would be used to determine by how many steps the device had to move, and in which direction.

Here, two flaws of the old firmware should be highlighted. First, due to a complicated conversion structure, it inadvertently used two different step counters for relative and absolute moves. This meant if the user mixed those two movement modes, the device would move to an arbitrary position and often crash or fall off the rail. Second, in addition to those two counters, it would also save the current position to the EEPROM after every move for no apparent reason. EEPROM is a type of non-volatile memory which supports only a limited number of write cycles, in the case of the ATxmega only 100,000 write cycles are guaranteed. During the diphenhydramine hydrochloride synthesis, most pumps would execute around 700-800 moves over the course of the entire procedure, and the pump servicing the liquid/liquid separator would execute exactly 1784 moves. This would mean that the pump would exceed the number of guaranteed write cycles after a mere 56 syntheses. It is not certain that the EEPROM will fail after exactly 100,000 cycles and it may well hold up significantly longer, but writing to the EEPROM after every single move still seems like an unnecessary shortening of the overall life span of the devices.

If the move was a very short one, and the overall number of required steps was smaller than the acceleration and deceleration phase, the routine would just issue a move over the entirety of required steps at the starting speed. Otherwise, the acceleration and deceleration phases were subtracted to yield the travel steps (see chapter 7.1.6), the travel speed in steps per second was calculated from the provided flow rate in $\mu\text{L}/\text{min}$, and the acceleration curve was prepared.

Since there is always some play in the system, partially due to manufacturing tolerances in the lead screw, and partially due to the system not being infinitely rigid, if the direction of a move was opposite to the direction of the last executed move, a small number of steps was added to the travel steps to compensate for this play. Those backlash compensation steps were chosen arbitrarily. An experimental determination of the required backlash steps was always planned, but ultimately there was never enough time for this experiment.

With all move parameters in hand, the move routine was started as described in chapter 7.1.6. The PWM frequency was set to the starting speed, the step counter was set to the first acceleration increment, and the DIR pin was set as required. Then the motor was enabled and the PWM generation started.

As mentioned in the previous chapter, the step counter would create an interrupt once the number of required steps had been taken, and the attached ISR would govern the further behaviour of the device. This ISR is structured as a state machine in order to guide the device through the motions in a concerted fashion. Figure 59 shows all states and transitions as a reference.

Regardless of the state, at the very beginning of the ISR, the PWM is disabled to avoid additional steps while the ISR is being processed. While the ISR would generally execute very fast, measurements showed that if the PWM continued throughout the ISR, a few additional pulses were sent, particularly at higher stepping frequencies. Thus, every move would be a few steps further than required. This effect is negligible for single moves, but over the course of hundreds or even thousands of moves the inaccuracy would add up and eventually throw off the positioning.

The old firmware used a similar approach, utilising a state machine inside the ISR, but it did not deactivate the PWM during ISR execution. When investigating the disconnection issues experienced with the old firmware, tests of several thousand random moves back to back were conducted without re-homing the pump in between. During those tests, the pump position was found to be completely off after some time, and the device would regularly crash against the base or fall off the rail.

Regularly re-homing the pumps would somewhat mitigate that problem, but precision by design was still preferable.

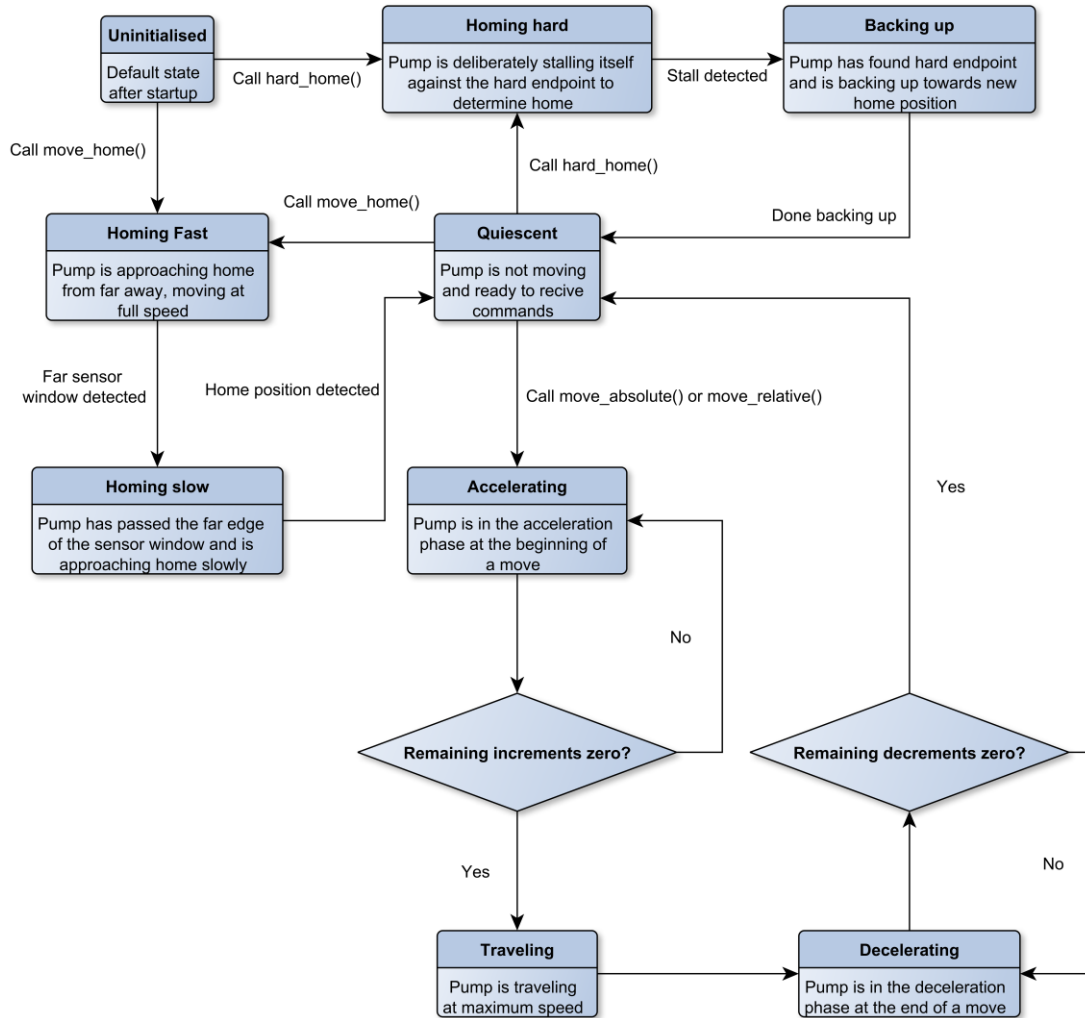


Figure 59: Full state machine of the pump.

Calling the absolute movement routine as described above would put the pump into the ACCELERATING state. In this state, the ISR would either initiate the next acceleration plateau as described in chapter 7.1.6, or if the number of remaining plateaus had reached zero, set the PWM to the travel frequency and the step counter to the travel steps and initiate free travel. This would transition the pump into the TRAVELING state.

The next time the ISR would be called when the device was in the TRAVELING state was when the free travel had concluded. Thus, the ISR would initiate the first deceleration step, and transition into the DECELERATING state. This state worked much like the ACCELERATING state except if the number of plateaus had reached zero. In this case, the ISR would deactivate the PWM and disable the motor, update the global position counter, and transition into the QUIESCENT state. In this state, the device was ready to receive another movement command.

Relative movement worked exactly the same way as absolute movement except for the determination of the required steps, which of course could now be done by directly converting the requested volume to steps. Other than that, the same sanity checks were executed, the move was initiated, and the ISR would transition through the state machine as described above.

Moving the pump to home worked slightly differently. The home position was determined by a magnet in the carriage and a Hall effect sensor in the base as described in chapter 6.1.4. When far away from home, the sensor would read close to 4095 ($2^{12}-1$ as the ATxmega ADC had 12 bit resolution). This reading would decrease as the carriage approached the home position, with readings at home of around 700-800 depending on the exact sensor and magnet used. The old firmware employed an algorithm that would move the pump by a small amount, then query the ADC value, followed by another small move, and so on. In another display of misguided sophistication, the width of every move was dynamically calculated based on the distance from home. While in principle this should allow for precise positioning, practically it would result in overly long ISR execution times with all the associated problems.

The alternative was to use the compare function of the ATxmega's ADC. This function allowed the user to store a value in a compare register, and if the ADC reading crossed this threshold in either direction, an interrupt would be generated. Thus, a move towards home could be initiated without counting steps and continued until the ADC interrupt signalled successful homing. To increase accuracy, a two-stage approach was adopted, where the pump moves towards home at the requested

(generally fast) speed until it reaches the edge of the magnetic field and the ADC reading starts to dip, followed by slow movement until the home position is reached.

When `move_home()` was called, the firmware would perform the usual sanity checks, followed by setting up the ADC to take readings continually and generate the compare interrupt once the compare threshold was crossed. The compare threshold was set to a value corresponding to the edge of the magnetic field. Then the movement was started directly at the requested move speed without acceleration. As the end point of the move would be determined by the sensor reading and not by counting steps, step loss is not a concern. A complete failure to move was not expected due to the generally low revolution speeds and the low loads and was indeed never encountered. The device would then transition into the `HOMING_FAST` state.

Once the ADC interrupt was triggered, the state machine ISR would decrease the move speed to a slow approach and set the ADC compare value to the home position. It would then resume movement, and transition into the `HOMING_SLOW` state. Once the home position was reached, the ISR would enter again, deactivate the PWM and disable the motor, and zero the step counter. The ADC was returned into a single read state where readings were only taken upon explicit request. This was done to avoid any unwanted interrupts to be generated and interfere with other movement routines. At the end, the device would again transition into the `QUIESCENT` state.

The home reading value used in this algorithm was stored in the device configuration (see chapter 7.1.9) and could be updated in one of three ways. An arbitrary value could be passed to the device while updating the device configuration. This mode is not recommended. Second, the pump could be manually positioned in a satisfactory way, followed by calling a function which would read the ADC and store the result as new home position.

This third and most elegant way was to use hard homing. This algorithm worked the same way as the normal homing routine, but rather than using an ADC interrupt to determine the end of the move, the TMC262's stall detection was used. Thus, the pump would move towards home, in the `HARD_HOMING` state, until a stall was

reported. Then, the pump would back up by half a millimetre to create a safety margin, read the ADC, and record the current position as home.

This procedure was found to work reasonably well. The stall detection was unfortunately not overly reliable, in particular it seems the abrupt hard stop against the base would sometimes elude the TMC262. In those cases, gently twisting the lead screw by hand would usually make the device recognise its situation and proceed to back up. The hard homing was usually called once at the start of a synthesis, to make sure all devices had correct home positions, and then from there on only normal homing was employed.

7.1.8 VALVE POSITIONING ALGORITHMS

For the valve, two movements were required, moving to an arbitrary position, and moving to home. As described in chapter 6.1.4 the positioning was achieved *via* a Hall effect sensor and magnets. Without a magnet present, the sensor would yield an ADC reading of around 2000 (half the maximum of 4095), and if a magnet is passed in front of the sensor, the reading would increase or decrease depending on the magnet direction.

Figure 60 shows a typical sensor response for a full revolution of the valve. The voltage is measured at the ADC input pin. The Hall sensor used in the devices has a quiescent voltage of 2.5V, and minimum and maximum voltages close to 0V and 5V, respectively, but the input circuitry on the board includes a voltage divider which halves the input voltage. The curve clearly shows the five “negative” magnets indicating the output ports 1-5, and the one “positive” magnet indicating output port 0, the home position. The curve also shows that the maximum readings of the positions are somewhat different, likely either due to varying quality of the magnets, or because some magnets are inserted further into the motor connector than others. In either case, though, the magnetic field appears to be reasonably symmetric. One of the peaks was fitted with an asymmetric double sigmoidal function using OriginPro 2016 (red line, see Figure 61 for fitting parameters), and the results showed that the peak was almost perfectly symmetrical.

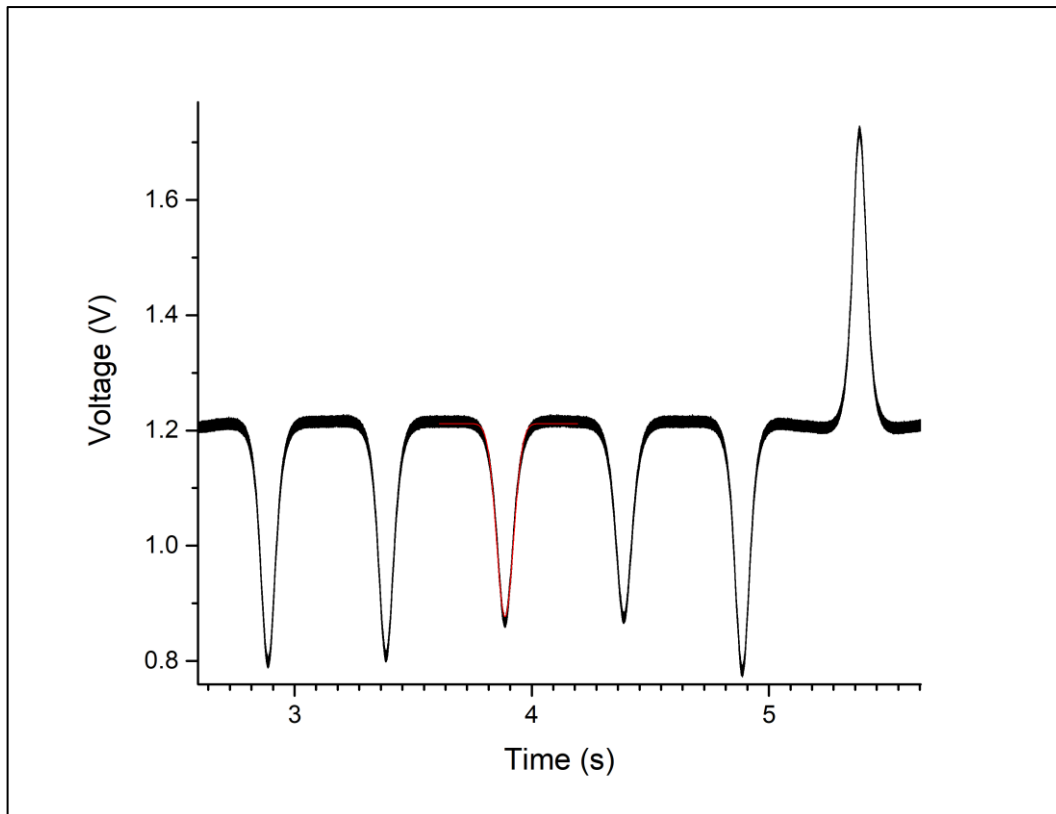


Figure 60: Hall effect sensor reading of one full revolution of a valve. The red line represents a double sigmoidal curve fitted to one of the peaks.

Thus, in order to find a valve position, the firmware had to find the maximum (or minimum) value of the sensor response. The old firmware employed an algorithm which repeatedly moved the motor by a small amount and polled the ADC, similar to how it determined the pump home position, a solution which was plagued by the same inefficiencies. Furthermore, the positioning would fail at random times. Extensive investigation into this issue by multiple persons could not pinpoint the reason for this failure. The failures seemed to happen more often the longer a valve had been in use, and some findings hinted at a heat related problem, but no single cause could be determined to any degree of certainty. Overly long ISRs were suspected to be causing trouble, but this theory was never confirmed.

Sceptical of such a step-and-measure approach, an algorithm leveraging the ADC interrupt was devised instead. The ADC compare value was set to an arbitrary threshold which was chosen to be roughly at half height of a typical response peak

(Figure 61). The ADC was set up to generate an interrupt if the measurement dropped below the threshold (or rose above for positive peaks), and the motor was set in motion. Once the interrupt was received, the step counter was zeroed, the ADC was configured to send an interrupt once the measurement rose above the threshold again (or dipped below for positive peaks), and the motor was set in motion again. Once this second interrupt was received, the motor was stopped and the step counter, representing the width of the peak, was read. Finding the centre of the peak was then a simple matter of moving the motor back by half the peak width.

This algorithm was implemented in the form of a state machine inside the step counter ISR, similar to the state machine for the pump. Figure 62 shows all states and transitions. The simplest case was a movement to the home position. After checking whether the device was error free and in the `QUIESCENT` state, the routine checked if it was already homed, in which case “actuation success” and “move done” were reported. Otherwise, the ADC interrupt was configured as described above. To determine the direction of the threshold crossing, the known direction of the home magnet (*vide infra*) was used, which was stored in the device configuration. However, the interrupt was not yet enabled to avoid a premature triggering if the valve happened to be close to the home position already. Then the devices would transition to the `ACCELERATING` state and perform the acceleration routine as described at length for the pump (see chapter 7.1.7).

Once the acceleration ramp was concluded, the ISR would set up a move at travel speed for 112.5% of a full revolution, the value of which was stored in the device configuration (see chapter 7.1.9). This was to avoid a deadlock where the valve would rotate perpetually if no magnet was detected for whichever reason. The additional 12.5% of a revolution were added as safety margin in case the device was precisely at home when the move started. 12.5% was chosen as it represents 1/8 of a revolution or a bit shift by three positions, which is significantly faster than a division by an arbitrary value.

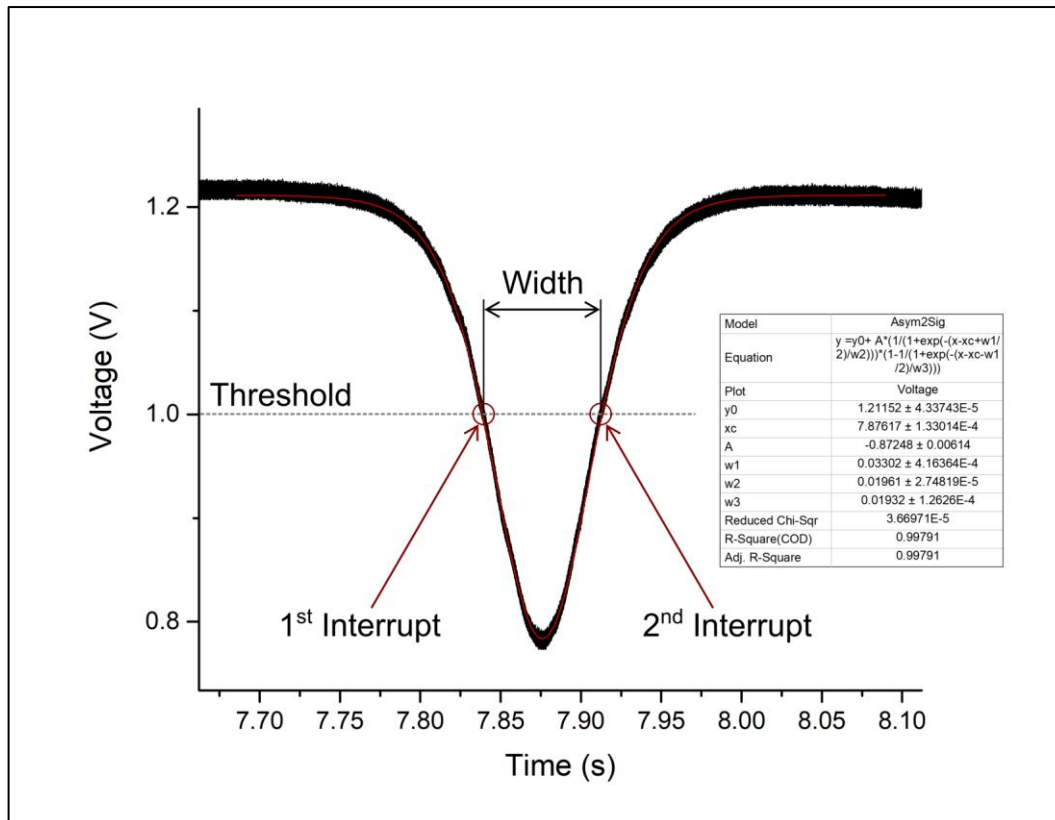


Figure 61: Detail view of the reading in Figure 60 showing the threshold and the two interrupts used in the positioning algorithm.

At that point, the ISR would also set up an auxiliary compare value for the step counter, which was set to a clearing distance. The compare match ISR, written in Assembly for performance reasons, would then turn on the ADC interrupt. As mentioned above, the purpose of this exercise was to avoid a premature triggering of the ADC interrupt if the device started close to home. For moving to home, this is more of an academic consideration, but for moving to a position (*vide infra*) this was a vital precaution as the valve would likely start any move from a position. If the ADC interrupt were active from the outset, it would trigger immediately and stop the move before it began. Thus, the ADC interrupt was set up but remained inactive until the valve had cleared the current position by a number of steps.

From there on the device, now in the `MAGNET_FOUND` state, would travel freely until the ADC detected the leading edge of a magnetic field. In this case, the ADC would be reconfigured to detect the opposite edge but would again remain inactive.

This was done to avoid an immediate triggering due to noise. The ISR would then zero the step counter, set up a move by another full revolution and set the auxiliary compare register to a number of blanking steps. This would ensure the sensor was well within the magnetic field and away from the noisy edges before the interrupt was reactivated by the compare match ISR. The device would then transition into the `EDGE_FOUND` state and continue movement at a slow speed.

Once the ADC detected the opposite edge on the magnetic field, the ISR would stop the motor, disable the ADC, and read the step counter to obtain the width of the magnetic field. The device would then slowly move back half of the width, transition into the `CENTRE_FOUND` state, and once the move was concluded turn off the motor and transition into the `QUIESCENT` state.

Moving to a certain position worked much the same way as moving to home, but for one difference. While there was only one home position, so the first magnet which was detected was assumed to be home, there were five positional magnets. Thus, before commencing the move, the routine would use the current position stored in memory to determine by how many positions the valve had to move. It would also determine which direction (clockwise or counter-clockwise) provided a shorter move. For moves by 180° it would arbitrarily choose a positive move (counter-clockwise when viewed from above).

Then, the move was initiated as described for moving to home. However, once the leading edge of a magnet was detected, the algorithm would check the number of positions to go, and if the current magnet is not the target, the algorithm would just skip it by remaining in the `MAGNET_FOUND` state, clearing the magnet using the auxiliary compare value, and looking for the next edge. This would go on until the target was reached, in which case the valve was moved to the centre of the position as described above.

RESULTS AND DISCUSSION

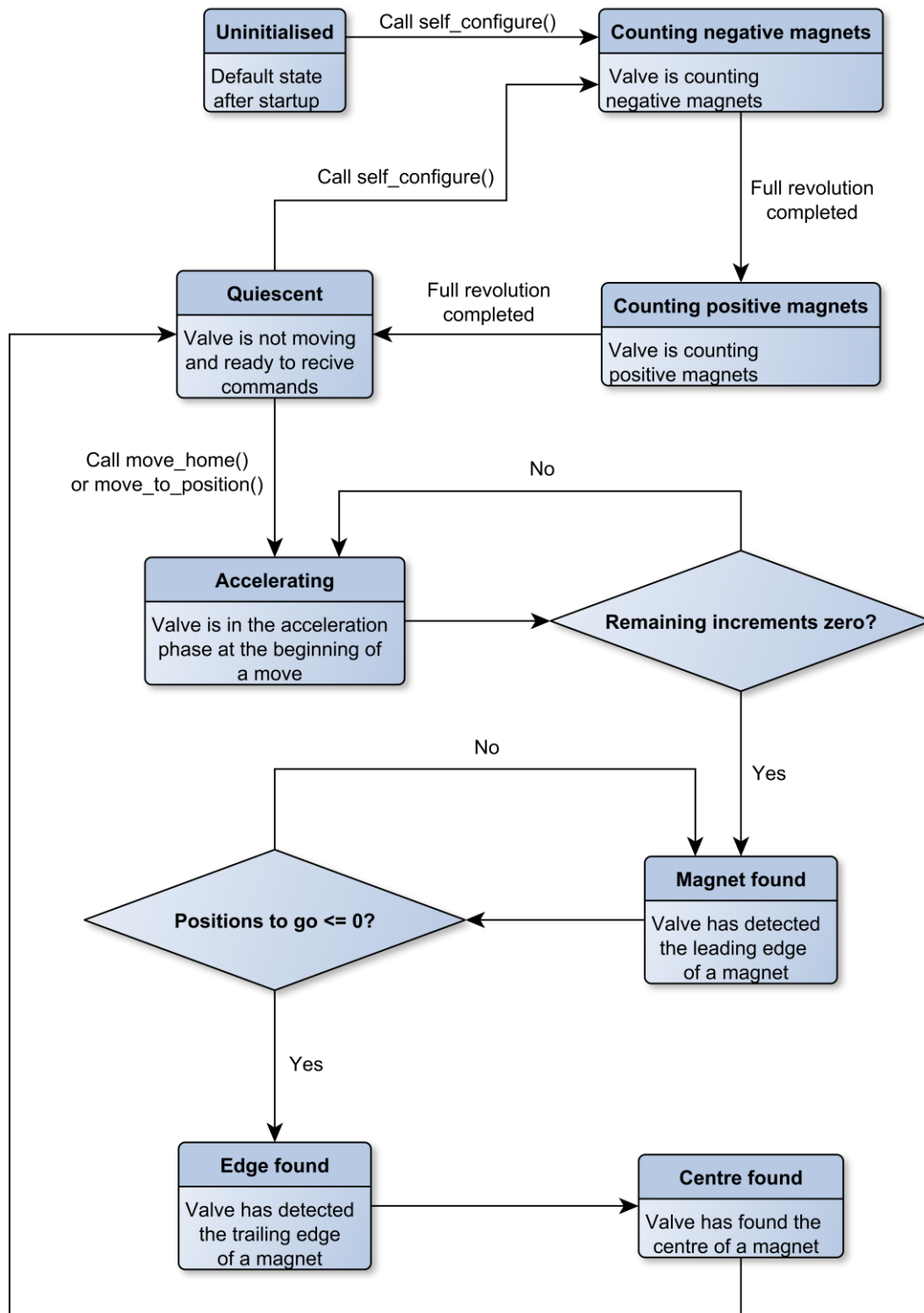


Figure 62: Full state machine of the valve.

As briefly mentioned above, the direction of the home position magnet was saved in the device configuration. However, evidently it had to be identified once in the first place. This could have been done by manually moving the valve to home and reading

the ADC, and additionally a workmanship standard was adopted demanding that all valves were assembled in the same way, i.e. the directions of the magnets to be the same for every device, but both solutions were unsatisfying.

Thus, a self-configuration routine was devised. In this mode, the valve would perform two full revolutions and count all positive and all negative magnets, respectively. The routine used the ADC interrupt as described above, and after registering one magnet it cleared it by a number of steps much the same as the other movement routines. Once the counting was done, there were, in essence, three possible cases: first, one positive and five negative magnets, which signified a positive home position. Second, one negative and five positive magnets, signifying a negative home position. Third, any other count, which signified a problem. In this case the magnet counts were reported back to the user and the device would flag an error (see chapter 7.1.9).

A commonly encountered error scenario was five magnets in one direction and two in the other. This happened if the valve started counting when centred on a position, in which case it would sometimes count this position twice. Other scenarios included six magnets in total, but more than one (or none) opposing the others. That was usually due to operator error during assembly. Similarly, fewer than six magnets in total were either due to a magnet missing, or due to demagnetising a magnet during assembly by use of concussive force. Any other error was most likely to be attributed to a faulty Hall effect sensor.

Self-configuration was normally done only once after programming the MCU. From there on, the home position direction was preserved in the device configuration, and re-running the self-configuration was unnecessary.

7.1.9 DEVICE CONFIGURATION AND ERRORS

The previous paragraphs frequently mentioned the device configuration, so it might be worthwhile to explain in detail what that is. In a nutshell, all important variables describing the current setup of the devices, such as syringe size for the pump, number of positions for the valve, step counts for the movement algorithms, and sensor reading thresholds, are stored in a structure called `globalConfig`. As pumps and valves require different information, there are two different configuration structures for them. During compilation, a pre-processor macro would determine which structure to incorporate depending on whether a symbol PUMP or VALVE was defined.

The configuration would be written to the EEPROM whenever it was changed. Practically, this happened rarely enough. Upon bootup, it would be read from the EEPROM and stored in RAM. The advantage of keeping it in the EEPROM was that it could be changed during runtime and those changes were preserved when the device was powered off. This made it a self-contained system, allowing the user to use it without having to configure it every time.

Similar to the device configuration, the devices also held a structure named `networkConfig` which contained all relevant information for the W5500 such as IP and MAC address. The network configuration was stored in the EEPROM as well and was read during bootup to be sent to the W5500.

Furthermore, one byte signifying the error state of the devices was stored in the EEPROM. Every bit represented one particular type of error. If an error condition occurred, the corresponding bit was set and the whole byte was written to the EEPROM in order to be preserved even if the device reboots. Multiple error bits could be set at any given time, and the error state could be queried by the PC, and errors could be cleared. The only error which could not be cleared by the user was the configuration error. This bit was set by default after programming the MCU and would only be cleared by supplying a device configuration. Otherwise movement routines could be called without having any of the required information, leading to undefined behaviour.

7.1.10 PYTHON API

Once the firmware was completed and preliminary testing had concluded, a colleague was tasked with writing an API in Python based on a specification sheet. After he delivered the first version, thorough testing and some further development was performed, and eventually the API was converted into an installable Python package. This way, any user can simply install the package and use the pumps and valves.

The API consists of a function providing a UDP keepalive, and three classes. The function starts a daemon thread which continually broadcasts the keepalive message into the specified subnet every half second and has to be called once at the beginning of any script using the API.

The first class is a generic parent class providing basic communication. Child classes for pump and valve provide methods for performing all practical operations required, such as moving the devices, writing configurations, or reading and clearing errors.

The entire package, including example files demonstrating the use as well as extensive documentation, is available from the Cronin Group upon request.

7.1.11 FIRMWARE TESTING

In order to thoroughly test the firmware and API, both a pump and a valve were subjected to a random move stress test. This test executed random moves back to back continually and logged the responses to a file. For the valve, random positions including home were performed. For the pump, moves to random absolute positions at random speeds were performed without moving to home. The values for position and speed were limited to sensical values, and new positions were required to be at least slightly different from the old position.

As the reliability of the network communication was a concern, the connection was monitored using Wireshark, and the Windows ping utility was used to send pings to the devices throughout the entire test.

RESULTS AND DISCUSSION

The pump successfully performed 11,875 moves before the test was terminated, which is more than any pump had ever managed with the old firmware before breaking down. Even without re-homing, the position was still accurate, no crash was observed. Out of 237,786 packages sent by the ping utility, 55 or 0.02% were lost. The valve performed 24,150 moves before the test was terminated, and lost 5 out of 8,755 packages, or 0.06%. Neither device failed. To put those numbers in perspective: during the diphenhydramine hydrochloride synthesis, pumps performed between 500 and 1800 moves, and valves performed between 500 and 2100 moves. Thus, both pump and valve were tested to around ten times the number of moves in a real-world application. The packet loss can't directly be correlated to the likelihood of a failure event since the TCP protocol will detect lost or corrupted packages and retransmit them as required. Every move consists of three network interactions: the PC sends a command, the device acknowledges receipt, and later on the device reports successful completion of the move. Based on that, a simplistic calculation would suggest that such a TCP retransmission would have to occur once or twice per synthesis which is perfectly acceptable.

In addition, the stability of an idle connection was tested. The old firmware would sometimes lose connection when the devices were just standing idle for several hours. Thus, the devices with the new firmware were connected to a PC and a TCP connection was initiated. The devices were then left idle over the course of a weekend, and the network was monitored using Wireshark. Neither pump nor valve dropped the connection.

Emboldened by those test results, the firmware was released for routine use. Since then, four synthesis platforms totalling over 50 devices were built and used, and all three syntheses described in chapter 8 were performed without any device failure.

7.2 THE SERIALLABWARE PROJECT

Aside from the pumps and valves which were using ethernet protocols for communication as discussed in the previous chapter, most other computer controllable lab equipment used a serial connection of some description. The most common ones are USB, RS232, and RS485. CAN bus would be another serial interface, but it is rarely encountered in laboratory equipment. All equipment used in this work was controlled *via* either USB, or RS232 in conjunction with an RS232-to-USB converter.

From a software perspective, most devices (and all devices used in this work) are recognised by the operating system as serial ports. Communication is achieved by sending commands as strings and receiving replies in the same way. There is no generally observed standard for the format of those strings, with every manufacturer defining their own communication standard specific to their products. The German User Association of Automation Technology in Process Industries (NAMUR) has published a suggestion for standardising serial communication with small lab equipment¹²⁸ which is more or less observed by many German manufacturers such as IKA®-Werke GmbH & CO. KG, Heidolph Instruments GmbH & CO. KG, VACUUBRAND GMBH + CO KG, or JULABO GmbH. Other equipment manufacturers such as Huber Kältemaschinenbau AG, however, developed their own standard instead. Generally, the communication protocol is described in the user manuals of the equipment. Thus, a user can implement their own control software if required, although most manufacturers will also sell turnkey software solutions.

In this work, several different classes of equipment sourced from a range of manufacturers were used. Using the proprietary software solutions was immediately out of the question as that would greatly increase software bloat and also would require obtaining permission from the developing companies. Thus, it was decided to implement Python drivers for every device and use them as a base layer for the overall control suite (see chapter 7.3).

Up until this point it was common practise in our research group to write specialised pieces of software to control lab equipment, which were usually hard-coded to serve

the immediate purpose at hand. Transferability between projects, and general code reusability were not commonly observed design objectives. This not only meant an unnecessary multiplication of effort, it also produced a host of mostly poorly written and undocumented code snippets which would often be re-used by new group members in a prime example of cargo cult programming.¹²⁹

Unsatisfied by the available solutions and driven by the need for reliable device drivers in order to build the synthesis platform, a code base for commonly used devices was created, which could then also be included in other projects developed in the group, and possibly made available as open source package to aid the wider scientific community.

The first device that was investigated was the IKA® RET control-visc hotplate stirrer (Figure 63). This computer controllable hotplate stirrer had been in use in our group for a while and featured in several automated platforms built at the time. A Python class was written using the pySerial package¹³⁰ to establish a connection to the device and allowed the user to read and write setpoints for the various functions like heating and stirring, as well as to read current process values.

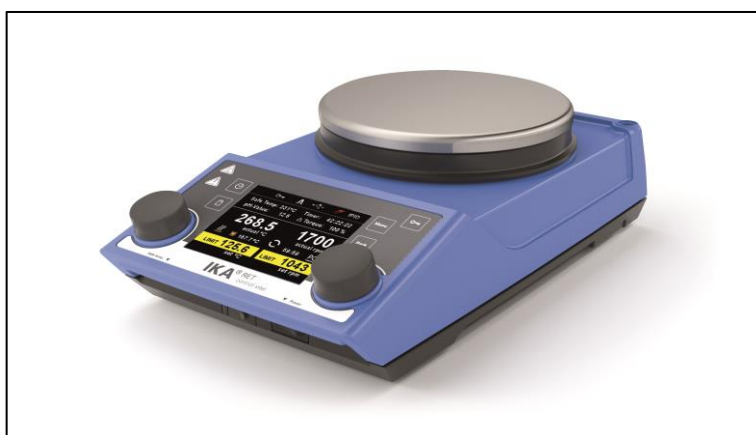


Figure 63: IKA® RET control-visc computer controllable hotplate stirrer.

This script was later refined by a colleague by moving the sending and receiving of commands into their own private methods. He also incorporated all communication codes into the class and created discrete methods for every function. Previously, a generic `read_setpoint()` method would take the respective function as an

argument, while in the new version, `stir_rate_sp()` and `temperature_sp()`, for example, were separate, distinct methods. The new structure also made use of the Python `@property` decorator, which allows a calling script to handle those methods as properties like so:

```
>>> hotplate.temperature_sp = 50
>>> print(hotplate.temperature_sp)
50
```

Where the first line would send the appropriate command to the hotplate and set the temperature setpoint to 50°C, and the second line would query the current setpoint and print it to the console.

This hotplate stirrer class was used in some early iterations of the synthesis platform, and in some other projects in the group. However, as further devices were added to the platform, it became clear that a more general solution was required.

As establishing a serial connection as well as sending and receiving commands are universal operations required for all types of devices, a generic parent class called `SerialDevice` was created to take care of those operations. For the settings of the serial port such as baud rate, bytesize, or parity, default values are defined as class attributes which can be overridden by device specific child classes. Similarly, default values for encoding the string to bytecode (UTF-8 by default) and for termination characters (carriage return – line feed by default) are defined as attributes.

When working with the IKA® RV 10 digital rotary evaporator (Figure 64) another requirement was found. The heating bath features a watchdog timer which triggers an error if the bath is in use, and no command has been received from the PC for some time. Thus, a regular keepalive function had to be implemented. This function had to exist in a separate thread for obvious reasons, but it also needed to be coordinated with regular command interactions. Thus, the entire communication was moved into a daemon thread. Henceforth, a method call in the main thread would place the appropriate command in a queue which passes it to the

RESULTS AND DISCUSSION

communication thread. There, the command would be dispatched, and any replies were put into another queue to be received by the caller in the main thread.



Figure 64: IKA® RV 10 digital rotary evaporator.

If no commands are waiting in the queue, the communication thread would execute a keepalive method. In the parent class, this method was merely a 100 millisecond wait, and any device specific child classes requiring a keepalive tick would override this method accordingly.

In order to pass a method call to the communication thread as described above, a decorator was created. In Python, decorators are wrapper functions that are applied to a function or method before execution. This wrapper function can be applied by using a special notation:

```
@spam
def foo(bar):
    # do stuff
```

Here, the decorator `@spam` above the definition of `foo()` means every time `foo()` is called, it results in a call to `spam(foo())`, where `spam()` is a function defined elsewhere.

In this project, an `@command` decorator was defined, which checks whether a method is called in the main thread or the communication thread. If it is called in the main thread, it is enqueued and passed to the communication thread. If it is called from within the communication thread, it is executed as is. Thus, in a device-specific child class, any method resulting in communication with the device must be decorated with `@command`.

Another feature implemented in the `SerialDevice` parent class was syntactic checking of responses using Regular Expressions (RegEx). Usually, when a command is sent to a device and an answer is expected, the general format of that answer is known. This might be as simple as expecting an integer when a setpoint is queried, or as complicated as expecting a response consisting of several characters signifying error codes, status messages, and queried values.

To ensure the answer is sensible, the user may define a RegEx pattern for the anticipated response, and if the answer does not satisfy the pattern, an error is raised. Furthermore, by using capture groups, the string returned by the device could be deconstructed within the communications thread, and an already sanitised version could be returned. Consider the IKA[®] RET control-visc hotplate stirrer for example. The device is able to report a range of values, including stirring speed, temperatures of multiple sensors, relative viscosity of the medium, and pH (if a pH probe is attached). When querying any of those numbers, the expected response is a floating-point number representing the value, and an integer representing the function (stir rate for example is represented by the number 4). The answer is consequently matched against a RegEx pattern `"(\d+\.\d+) (\d) \r\n"` (read: one or more digits, a decimal point, one or more digits, a space, and another digit, followed by a carriage return and a line feed). Everything within parentheses, i.e. the two numbers, is captured, and the calling method will return a list of the two numbers as strings.

RESULTS AND DISCUSSION

The whole project was named SerialLabware and converted into an installable Python package. Subsequently, classes for ten devices from six different vendors were implemented and used successfully in the synthesis platform. The source code is available from the Cronin Group upon request.

At the time of writing, curatorship of the project was being transferred to a colleague for continued development. Several changes are planned, most importantly the removal of the `@command` decorator. This structure, while perfectly functional, was found to be confusing for other users, and in hindsight it is not entirely necessary, so it is scheduled to be removed. We also hope to be able to publish the project in an open source format to enable other researchers to more easily interface with their lab equipment.

7.3 THE CHEMPILER

7.3.1 MOTIVATION AND REQUIREMENTS

At the outset of the project, it was quickly realised that, in order to fully utilise the modularity of the hardware, any software solution controlling the platform needed to be equally modular and adaptive. If a control suite is tied to one particular implementation of the platform, this effectively hamstrings the modularity and reconfigurability of the platform.

From this, two requirements arise. First, the addition of new modules must be possible with as little change to the overall software as possible. Adding more Backbone units, swapping hardware such as stirrers or chillers for equivalent devices from other vendors, or implementing entirely new unit operation modules should all be readily doable without major coding effort. Secondly, the code must have a level of awareness of the physical configuration in order to direct commands to the appropriate modules. To give two examples: if the user wants to move a given chemical from, say, a feedstock bottle to the reactor, the system must know the location of those two vessels as well as the fluidic connectivity of the pumps and valves in between, so it can orchestrate the move and issue the right commands in the right order. If the user then wants to stir the reactor, the software should have knowledge of the make and model of the stirrer in order to send the appropriate low-level commands, and it should have knowledge of the communication protocol and address of the stirrer servicing the reactor in particular, so commands can be dispatched to the correct end point.

The awareness of the physical structure of the platform is also a great boon in terms of usability. It enables an operator to issue high-level commands such as “move a chemical from A to B” or “stir and heat a vessel” which reflects the way an experimental chemist is thinking about a synthetic procedure. This lowers the barrier of entry as a user doesn’t need in-depth knowledge of the inner workings of the system in order to successfully script and execute a synthesis.

7.3.2 THE CHEMOS AND “THE SCRIPT”

Based on this assessment, Dr. Gerardo Aragon-Camarasa set out to program a software solution dubbed “ChemOS”. Its central module was a message passing service implemented in Erlang, a programming language developed for telecommunication applications.¹³¹ This ChemOS core would then connect various processes written in Python, which would perform tasks such as controlling hardware, scheduling operations, or parsing a rudimentary scripting language.

The development of the ChemOS suite was delayed repeatedly, so as a workaround, a pure Python solution referred to simply as “The Script” was implemented. This consisted of a single Python script for every synthesis, containing hardcoded implementations of all the required operations. Every script imported drivers for the pumps and valves (see chapter 7.1.10) as well as for the hotplate stirrers and other labware using serial communication (see chapter 7.2). The physical setup of the platform was captured in several large dictionaries which were maintained manually. This regularly led to problems when entries did not match up, and generally was a laborious and inconvenient solution. Variables for all volumes, times, temperatures and other parameters were also defined at the start of the script.

A number of functions were defined, most importantly an `add_from_to(source, destination, volume)` directive which would find the provided source and destination flasks, work out a fluidic pathway based on the aforementioned dictionaries, and compile a series of atomic pump and valve commands which would eventually result in moving fluid across the Backbone as described in chapter 5. Other helper functions included a waiting routine which would periodically log a status message to assure the user the script is still running, and the functions for liquid/liquid separation as described in detail in chapters 6.2.3 and 8.1.3.

All this setup effort was then followed by a lengthy if/elif structure providing the actual synthetic instructions. The idea behind this state machine was to enable a user to run the script from any arbitrary point onwards and skip steps at will. A variable `reaction_stage` kept track of the current code block. Every elif statement tested

`reaction_stage`, and at the end of every `elif` statement `reaction_stage` was reassigned to point to the next code block.

This monolithic architecture goes to show that “The Script” was only a temporary solution as chemical development (see chapter 8) had to progress. At some point, the entire diphenhydramine hydrochloride synthesis (chapter 8.1) was encoded in one script totalling over 2,000 lines. It successfully performed the synthesis, but it was unwieldy and near impossible to maintain. A better solution was sorely needed.

At this point, the status of the ChemOS was roughly as follows. The architecture of the platform was captured in an XML file compiled from a number of XML source files using `xacro`,¹³² an extension to the XML standard allowing the definition of macros inside an XML file. In a nutshell, a number of primitives such as a flask, a pump, or a tube were defined as individual XML files. The relationships between the individual nodes (flasks, pumps, valves) and edges (tubing) of the graph were defined in additional XML files. `Xacro` would then compile those source files into one XML file describing the platform. One `Chempiler` module would then parse this XML file into a Python graph object using the `NetworkX` package.¹³³

There are a number of issues with this approach. First, the user is required to carefully maintain the XML source files and ensure everything matches up, otherwise nodes that are supposed to be connected will instead have two outgoing edges to nowhere. As there was no graphical representation available at that point, visualising the configuration and checking for errors was near impossible. Every small edit had the potential of breaking the entire file. Also, parsing the custom XML structure into a `NetworkX`-graph was a complicated process requiring a dedicated Python module. As `NetworkX`-graphs are, in essence, dictionaries of dictionaries, the parser was full of hard-to-follow references to values inside multiple nested dictionaries. This led to numerous bugs which were extremely hard to pinpoint and fix.

Instructions for operations were provided in a pseudo-scripting language containing “S” or “P” as keywords to denote sequential or parallel operations; the name of the software module in question; the name of the operation, and any parameters. Those

instructions were parsed by a scheduler module in Python, and then passed to the Erlang core.

With the platform architecture and the instructions in hand, yet another Python module handled liquid movement. For every `move` instruction, NetworkX was used to determine the shortest pathway. A stack of individual pump commands was then compiled and dispatched to the devices. Control of anything other than pumps and valves was not initially implemented.

ChemOS introduced many important ideas and also showed many potential pitfalls, however, unfortunately it never came to fruition. As it became clearer that it did not meet the requirements of the platform it was decided to abandon the ChemOS project and instead re-write a control software from scratch, drawing on the lessons learned from ChemOS.

7.3.3 DRAFTING A SPECIFICATION FOR THE CHEMPILER

Having learned from the mistakes made with the ChemOS, a coherent plan was drafted. To that end, the features implemented in ChemOS were evaluated and classified as useful/desirable or useless/undesirable. Based on that assessment, a number of requirements were formulated.

First, the entire project should be written entirely in Python 3 to enable independent development and maintenance of the application as well as greatly simplify debugging. No need for another language could be identified as all required operations (*vide infra*) could be comfortably implemented in Python.

The separate files for the physical layout and the synthetic operations were deemed a useful feature, as this would largely disconnect the scripting of the procedure from the layout of the platform. Operations could be specified similarly to a classical, published experimental procedure in prose, describing the manipulations involved without over-specifying the hardware. It was also decided to retain NetworkX for handling the graph, as it is an established Python module offering all required capabilities. Information regarding serial devices such as hotplate stirrers should

henceforth be contained in the graph, allowing the user to address them *via* the vessels they are attached to.

From the previous experiences with “The Script”, volume tracking was found to be a highly desirable feature. Previously, when quantitatively transferring the contents of a flask, the liquid contents of the vessel at that point were manually calculated, which was understandably prone to error. By storing the volume of each flask as a node property in the graph and updating it every time volume is added or removed, a “move all” operation could be implemented, and the aforementioned potential for errors could be eliminated. While this automated dead reckoning approach is still far from ideal, and at some point, a sensor based active feedback system will have to be implemented, it is still preferable to the alternative.

The control software was also intended to produce as much diagnostic data as possible in order to aid with trouble shooting. Besides, with the advent of machine learning and data mining techniques in chemical research,¹³⁴ detailed logs of successful and failed experiments might one day become a valuable resource. The ChemOS featured some basic logging capabilities on the Erlang side and also produced some independent debug print on the Python side, but on both sides, logging was more of an afterthought. With “The Script”, good experiences were made with Python’s inbuilt logging module, and a hierarchical logging approach. Low-level information capturing almost each and every hardware operation was directly logged to a file, while high-level, user relevant information was also displayed on screen. Thus, this approach was retained.

Based on those requirements, a rough architecture for the control suite was drafted (Figure 65). For reasons of clarity, it was not called “ChemOS”, mainly to avoid confusion. Also, the operation of the application bears little resemblance with an operating system (OS). Instead, the operation of this new control suite resembled the operation of a compiler. Modern high-level programming languages contain logical operations and procedures which are agnostic of the processor architecture. A compiler toolchain parses the text a programmer writes and produces machine code which runs on one specific processor. Programs can therefore be compiled to run on many different computers without having to be re-written every time. In the

case of the synthesis platform, the physical setup would represent the processor architecture, and the synthetic operations form a crude, high-level programming language. Consequently, the individual commands being sent to the devices correspond to machine code in computer science. Thus, and given that the working title for the synthesis platform was “Chemputer”, it was decided to call this software the Chempiler, a portmanteau of “chemical compiler”.

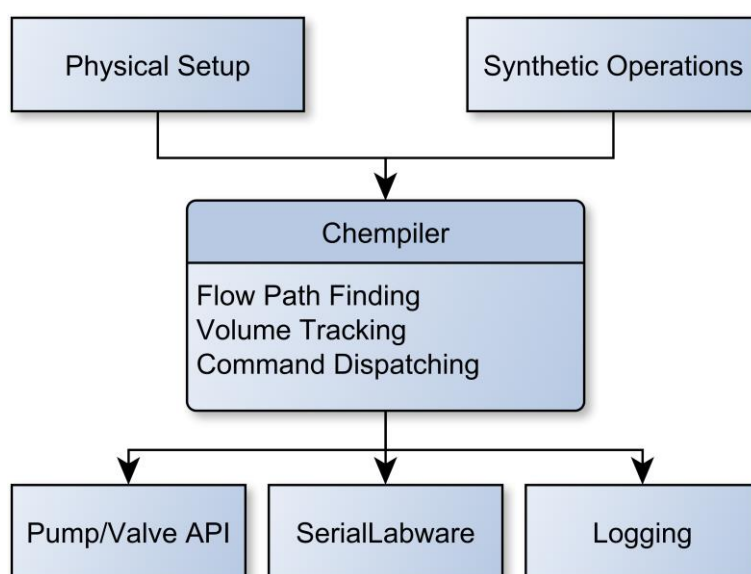


Figure 65: General architecture proposed for the Chempiler.

7.3.4 BUILDING THE CHEMPILER

Based on the aforementioned specifications, Graham Keenan started building the Chempiler and provided me with a basic implementation (Figure 66) which I could test and expand. The XML and command file formats and the respective parsers were initially retained to facilitate development. Figure 66 shows an overview of the structure of the first Chempiler implementation. The graph object obtained from the XML file and parser was passed to a setup script. This script extracted all references to devices, i.e. pumps, valves, and all serial devices such as hotplate stirrers. Using the pump and valve API (see chapter 7.1.10) and the SerialLabware package (see chapter 7.2), it instantiated objects for every device and stored those objects in the graph.

The list of operations was passed to a command dispatcher which used the graph object to send the individual commands to so-called executioners. Those are modules dedicated to one class of device and provide an intermediate layer which can take a range of device drivers (for example for different models of hotplate stirrers) and expose uniform methods to be called by the command dispatcher.

The executioner layer serves two purposes. First, as described above, it homogenises the interfaces provided by different device drivers. Thus, adding another model of any device is, at best, as simple as providing driver software and instantiating the device in the setup script. At worst, a few more lines of code have to be added to the executioners. In either case, a high-level instruction such as “start stirring” remains valid, which means the synthetic operations don’t have to be changed. The other purpose is to allow simple addition of entirely new classes of devices. If a user wants to implement a device not yet included in the Chempiler suite, they can develop an executioner for it, unit test the code, and add it to the command dispatcher by just adding a few more lines of code. Admittedly, a mechanism by which new executioners and their methods are automatically recognised and included would be preferable, but robustly implementing such a mechanism was deemed too complicated for too little a return. It was reasoned that entirely new device classes would only be added infrequently, so this manual approach was deemed sufficient.

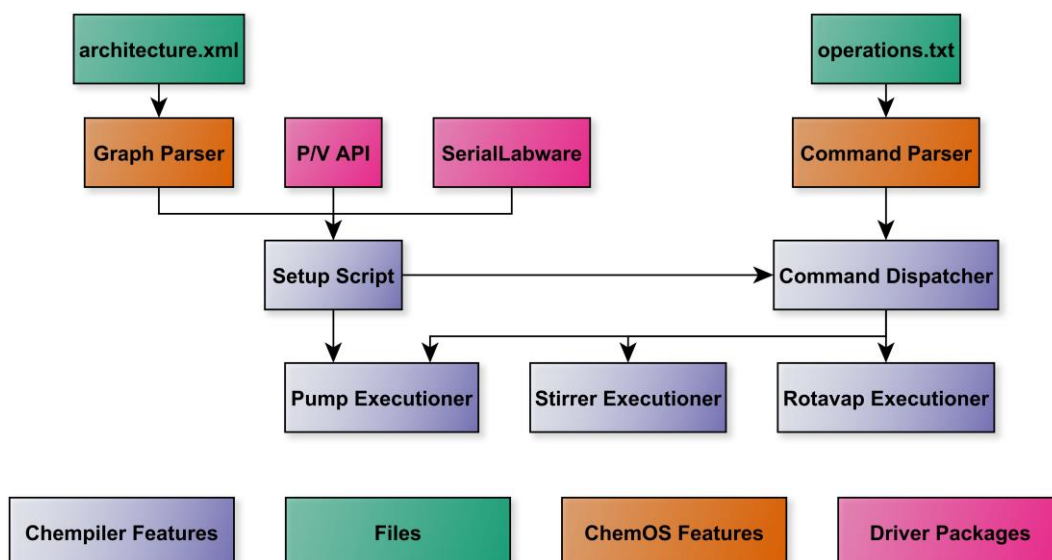


Figure 66: General architecture of the first Chempiler implementation. Files containing the physical layout of the platform (`architecture.xml`) and the synthetic operations (`operations.txt`) are parsed by modules adapted from ChemOS. The graph object representing the platform is used in conjunction with the device drivers to instantiate all devices. A command dispatcher used the graph object to direct the operations to the respective executioners.

7.3.5 MOVING LIQUIDS

The most important executioner was the pump executioner, as it included all the routines for moving liquid around the platform. The algorithm employed by ChemOS for this purpose was unclear to the author and was also found to produce incorrect movements on occasion. Thus, a new algorithm was developed. Figure 67 shows the first part of the algorithm. The command dispatcher would use the graph object to obtain a path between the source flask and the destination flask. A path length of 0 would mean the source and destination are connected to each other directly without liquid handling equipment in between, which would raise an error. A path length of 1 signifies that both source and target are connected to the same valve. In this case, moving liquid is a simple case of switching the valve and moving the pump.

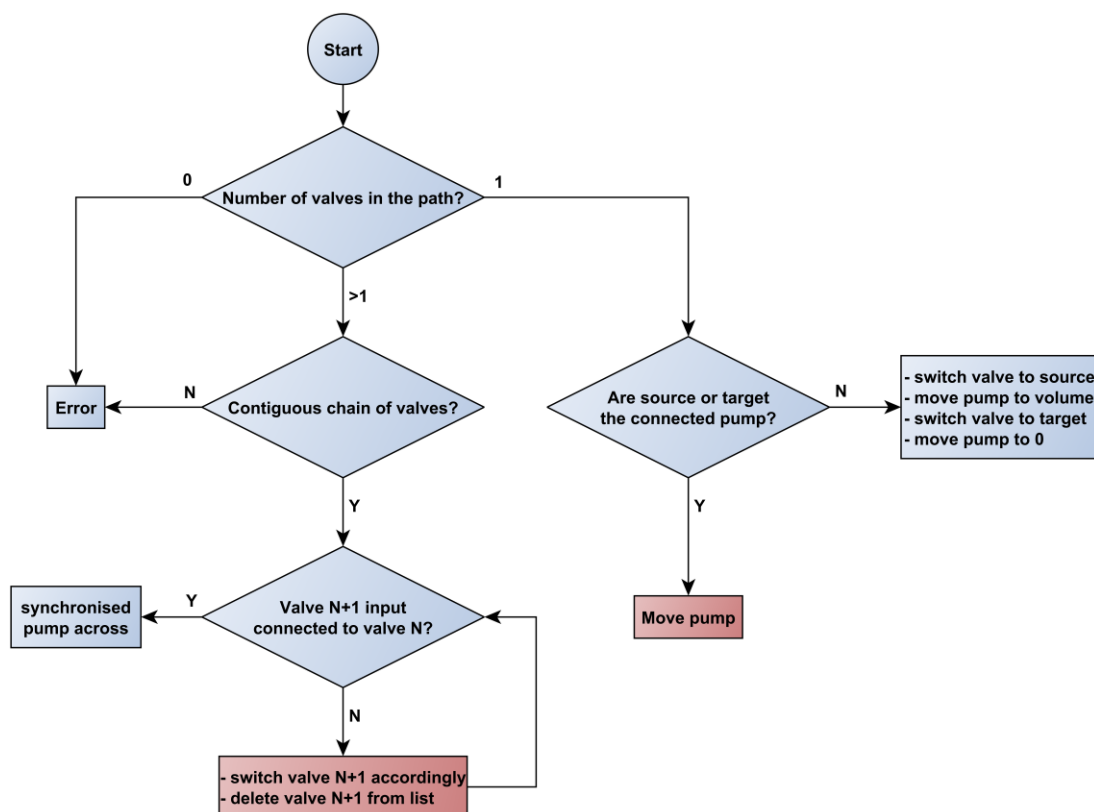


Figure 67: Flow chart detailing the new liquid transfer algorithm. Cases in red were deemed unlikely or not useful and therefore not implemented.

A path length greater than one means material is to be transferred along the Backbone. As a quick sanity check, the algorithm makes sure that all nodes in the path are valves, otherwise the scheme would not work. Then it initiates the movement routine depicted in Figure 17.

Two cases were identified that are, in principle, possible but not useful. One case is where the source or target is a pump. This case was dismissed as not useful, as just filling a pump doesn't serve any immediately obvious purpose. Another case identified was a tree-like architecture, where the input of one valve is connected to an output on another valve. Logically, this can be resolved by switching all valves to the appropriate position, except for the "root" valve which is connected to the pump. Then, the rest of the move routine can be executed as normal. A tree architecture may in theory be useful because it multiplies the number of reagent inputs. However, practically, problems associated with cross-contamination and dead volumes greatly outweigh the advantages. Thus, a decision was made against using such setups, and

therefore no need for implementing the associated move logic was recognised. A case not represented in Figure 67 is if source and target are identical. Initially this case was dismissed as not useful, however, it was later found that for viscous or reactive compounds, priming the syringe by aspirating a small amount and dispensing it back into the same vessel could improve accuracy, so a check against that case before entering the decision tree was later added to the function.

The synchronised pumping routine (see Figure 17) used in the case of path lengths greater than one had been in use for a long time and was found to work very well for volumes smaller than the syringe volume. For larger volumes, initially the movement was just repeated until the entire volume was transferred. Unfortunately, this was very inefficient, and moving larger volumes for example for cleaning purposes took a very long time.

Therefore, a staggered movement routine was implemented (Figure 68). Once the first pump in the path had passed its contents on to the next pump, it aspirated the next bolus in the next cycle, and passed it on, and so forth. This intensification made maximum use of the hardware and reduced the time required to move large volumes significantly. From a programming standpoint, the beauty of this approach was that it followed a simple, two-stroke rhythm, and could therefore be implemented in just a few lines of code.

RESULTS AND DISCUSSION

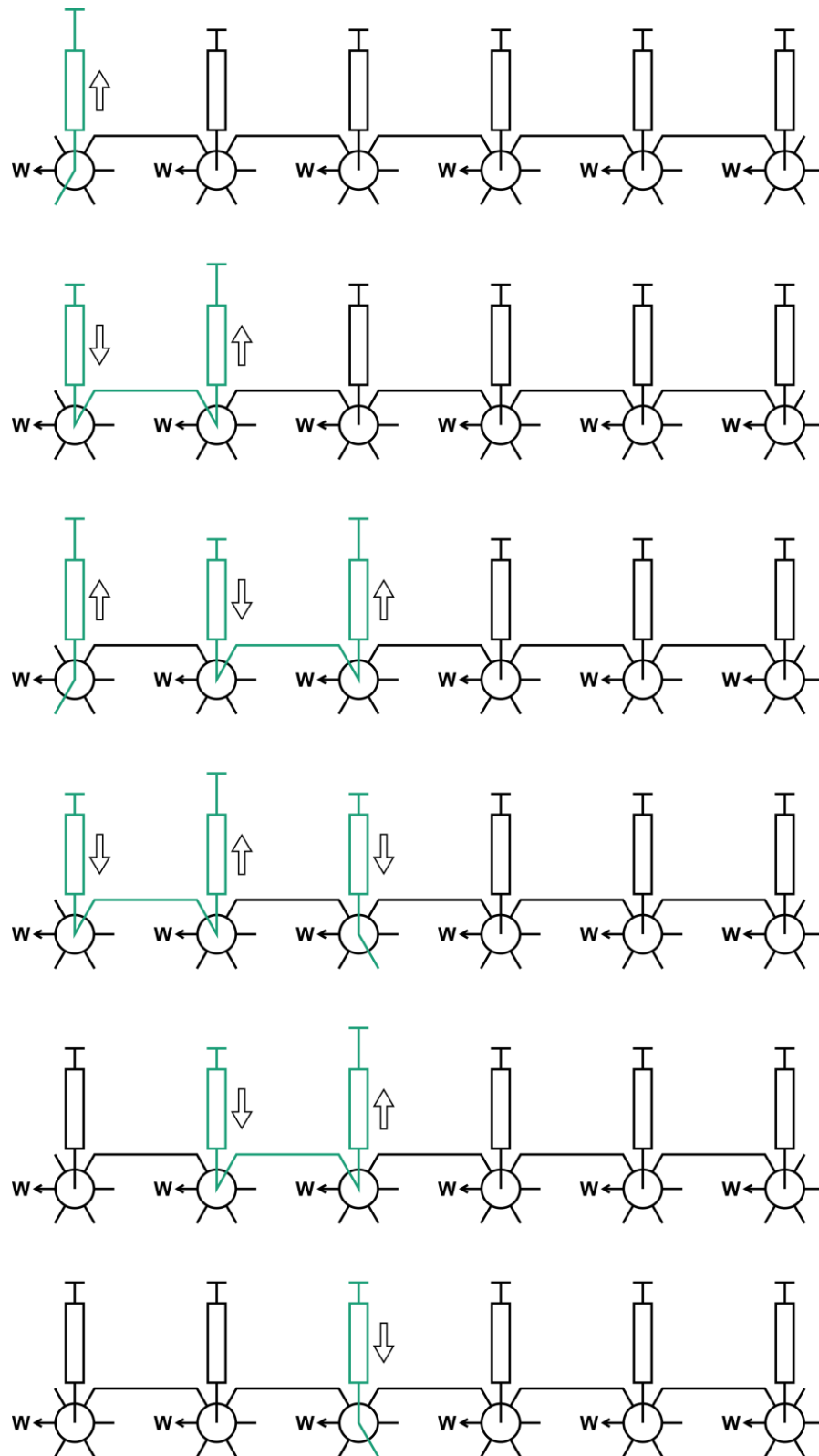


Figure 68: Staggered movement scheme. Once the first fill of the first syringe has been passed on to the second syringe, the first syringe refills itself again in the next cycle, and so forth. In this particular case, two full syringes were transferred in six movement cycles as opposed to eight required cycles if a normal move (four cycles in this case) is repeated twice.

7.3.6 FROM XML TO GRAPHML

Having served its purpose as proxy for testing, the ChemOS XML was now due to be replaced. Unsurprisingly, drawing, editing, and encoding graphs is not an entirely unprecedented problem. Numerous data formats dedicated to representing graphs exist, and an even greater number of editing programs are available, many of them as freeware.

After reviewing some options, GraphML was chosen as the new format to represent the physical setup of the platforms. GraphML is an open-source standard based on XML.¹³⁵ The NetworkX module can import GraphML files directly, and a range of graph drawing programs can export to GraphML. yEd¹³⁶ was chosen to draw and edit the architectures, a free graph editor which natively works in GraphML. This allowed the user to quickly draw a graph that represents the physical setup of the platform, visually verify its accuracy, and import the file directly into the Chempiler.

Figure 69 shows an example of a graph created with yEd. The entire graph of a six Backbone unit platform is too big to be usefully displayed on an A4 page, so one Backbone unit is enlarged to show the details. The size is not an issue during normal use, as the graph is only handled on screen where zooming and panning is possible. The nodes are pictures representing the individual modules, so even without the labels they can be identified at a glance. The images are purely for the benefit of the user and have no relevance once the GraphML file is imported into the Chempiler. All tubing on the physical platform is labelled with clip-on labels displaying a number and a corresponding colour as can be seen in Figure 51, and in the graph, the edges are automatically coloured to match that label. This way, the graph file can be compared against the real platform at a glance as well. Those optical features, while functionally superfluous as far as the program is concerned, were found to tremendously improve the user experience.

RESULTS AND DISCUSSION

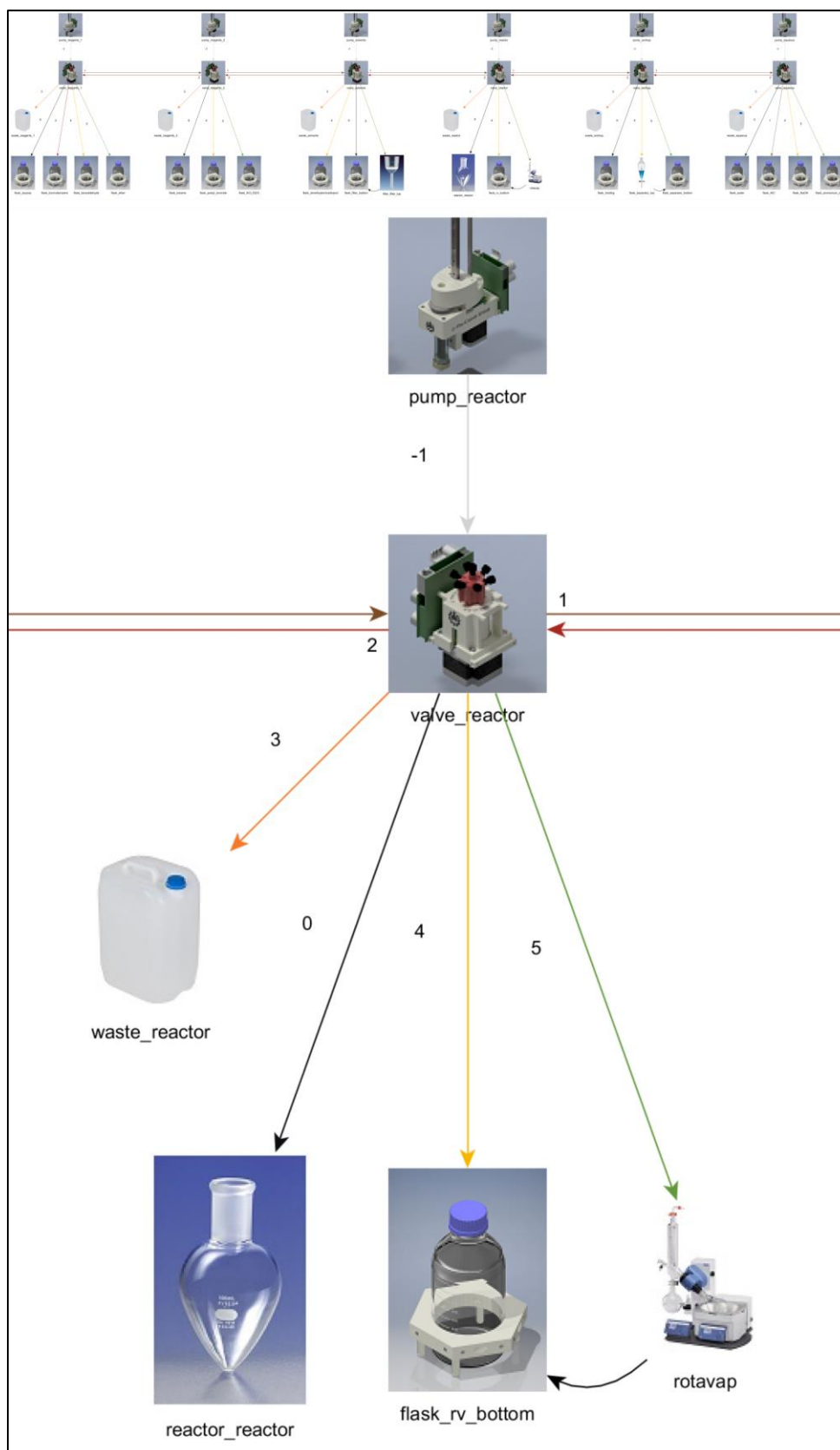


Figure 69: Example of a graph used in the synthesis of diphenhydramine hydrochloride. Top: overview over the entire graph. Bottom: Detail of one Backbone unit.

RESULTS AND DISCUSSION

All graphs created with yEd are directed graphs. While converting them to undirected graphs only requires a single line of Python, the advantages of the directed edges, as long as strict rules were applied, was quickly realised. In particular, it helps with distinguishing pumps and valves from vessels and flasks while orchestrating moves. While this could be performed on an undirected graph just as well, only having to iterate over successors or predecessors rather than all neighbours makes the code more efficient, and most importantly, more legible.

The rules for the number and direction of the edges are as follows:

- Pumps may have exactly one edge going out
- Valves may have one edge coming in representing the central inlet
- Valves may have one edge going out per outlet port, those edges have a “port” property between 0 and 5
- If two valves are connected to each other outlet port to outlet port, two edges must exist between them, where the “port” properties of each edge represent the port from which they originate
- Any other nodes may have only edges coming in, except for:
- Filters, separators and rotary evaporators may have one edge going out connecting it to their respective “other end” (i.e. distillate flask for RV, bottom outlet for filter or separator)
- Other than that, no two nodes that are both not valves may be connected

In addition to those rules, no two nodes may have identical names. yEd assigns unique labels to every node and stores the name as a node property, but for numerous reasons the Chempiler accesses the nodes by name. Thus, two identically named nodes will create undefined behaviour.

yEd allows an arbitrary number of data fields (called “properties”) to be assigned to nodes and edges. By default, both have two fields “URL” and “Description”, both of which are not used. For use with the Chempiler, the following custom properties are defined:

Table 2: Custom node properties of the yEd graph.

Name	Type	Default Value	Description
class	Text	none	Used by the Chempiler to distinguish pumps, valves, flasks, etc.
max_volume	Decimal	-1	Used by the volume tracker to issue warnings if a vessel is overfilled. <code>max_volume</code> of pumps is also used for the movement routine.
current_volume	Decimal	-1	Used by the volume tracker.
Chemical	Text	none	Reserved for future use.
IP_address	Text	none	Address of a pump or valve.
serial_device_1_type	Text	none	Make and model of serial devices.
serial_device_1_port	Text	none	Serial port of the device.

Table 3: Custom edge properties of the yEd graph.

Name	Type	Default Value	Description
volume	Decimal	-1	Dead volume of the tube. Used for priming the tube.
chemical	Text	none	Reserved for future use.
port	Integer	-1	Valve port the tube is attached to (0-5).

The node properties for serial device type and port exist three times to allow more than one serial device per node (e.g. a rotary evaporator consists of the evaporator itself, and the vacuum pump). The corresponding properties “`serial_device_2_type`”, “`serial_device_2_port`”, “`serial_device_3_type`”, and “`serial_device_3_port`” are omitted in Table 2 for reasons of brevity. Should any node require more than three serial devices, additional fields can be added. The setup script that parses the graph

and instantiates serial devices will iterate over all serial device fields regardless of the number.

Data fields must be populated where applicable, i.e. pumps and valves require an IP address, and all attached serial devices have to be listed. Edges require the ports except for those between pumps and valves. The Chempiler will raise an error if required data is not present, but the user is encouraged to check all nodes and edges prior to running a synthesis.

The most convenient way to construct a graph for a new synthesis, or even a new platform, is to edit an existing graph. This minimises the potential for error, and in most practical use cases the only information that changes between different syntheses is the name and placement of the reagent.

There are some physical devices which have two connections to the Backbone. Those are: the rotary evaporator (evaporation flask and distillate flask), the separator, and the filter (top and bottom inlets). The two connections are separate nodes in the graph, but they are mapped to each other. In yEd, the “top” and “bottom” nodes are connected with an edge. When the graph is loaded into the Chempiler, this edge is removed and the mapping is stored as node properties inside the graph object.

When the graph is loaded by the Chempiler, it is passed to the setup script. This module replaces the unique labels assigned by yEd with the node names in order to simplify the graph for further manipulations. It then iterates over the nodes, finds all pumps and valves, instantiates them, and stores the pump or valve object as node property in the graph object. This process is then repeated for serial devices. If one serial device, for example a recirculation chiller, is attached to more than one node, it is not instantiated again but rather the existing object is attached to both nodes. As last step, the script finds devices with two connections and sets them up as discussed above. The resulting graph object is then passed to the command dispatcher for further use.

7.3.7 THE CHEMICAL ASSEMBLY LANGUAGE CHASM

ChemOS featured a limited scripting language to spell out synthetic instructions, and an improvised parser. While this was good enough for initial testing, the limitations of this approach were quickly realised. The most sensible solution at this point would have been to implement a top level Chempiler class which exposes all operations as methods, so a user could start a Python file, import the Chempiler, and script a synthesis using the full extent of the Python interpreter's capabilities. However, having a dedicated "chemical programming language" was seen as desirable by some. Thus, a rudimentary scripting language was developed based on the current capabilities of the system. This scripting language turned out to superficially look similar to assembly code (commonly abbreviated as "ASM"), thus it was dubbed "chemical assembly language" or ChASM.

ChASM was designed to be a rudimentary scripting language acting as a thin layer on top of the Chempiler operation. It supports the definition of variables and re-usable functions, as well as loops, and comments. The following explanations use the Backus-Naur form (BNF)¹³⁷ to describe the grammar of the language. The basic element is the instruction, which has the following syntax:

```
<instruction> ::= <PS_TOK> <OPCODE> "(" <arg_list> ")" ";"
                | <PS_TOK> <OPCODE> "(" ")" ";"
```

<OPCODE> refers to a keyword recognized by the Chempiler as a module operation, which is associated with a list of arguments <arg_list> specific to the command. A Markdown file detailing all currently implemented commands can be found in the Chempiler documentation and in appendix IV. <PS_TOK> simply refers to the uppercase letters "P" and "S" for "parallel" or "sequential" operations, respectively. Instructions marked with an "S" are carried out in sequence. A number of consecutive instructions marked with a "P" are carried out in parallel by spawning an individual thread for every operation. Before the next "S" operation is carried out, the Chempiler waits for completion of all parallel operations. This notation was taken from the original ChemOS instructions, but unfortunately, this approach was found to be fundamentally flawed. First, issuing multiple commands to the same piece of equipment simultaneously from different threads usually results in undefined

behaviour, or outright crashing. Furthermore, parallelism in the scripting language demands at least rudimentary control structures such as semaphores to be of any use. Lastly, adding parallelism introduces all the challenges associated with concurrent programming, which directly counteracts the intended purpose of providing a simple interface for scripting syntheses. While at some point, parallelism should be implemented in some way in order to improve the efficiency of the platform, for the time being strictly sequential operation is mandatory.

To improve code legibility, maintainability and reuse, variables can be defined. Variables may be strings or numbers. The syntax for variable assignments is as follows:

```
<variable> ::= "DEF" <name> "=" <value> ";"
```

The language does not allow reassignment of a variable once it is defined. However, since the script is parsed once at the beginning of program execution and all logic happens within the Chempiler, the utility of reassignments is severely limited and did not warrant the effort of implementation.

Since many operations are repetitive, the definition of functions was deemed useful. The syntax is as follows:

```
<function> ::= "DEF" <name> "(" <arg_list> ")" "{" <body> "}" ";"
           | "DEF" <name> "(" ")" "{" <body> "}" ";"
```

Where <body> may contain instructions and other functions. Any variables and functions used within the body have to be defined beforehand.

Loops are useful for repeated operations. ChASM only provides a simplistic version of a “for” type loop, where the content of the loop is executed a number of times stated in the loop definition (i.e. “FOR (3)” executes it three times). The syntax is as follows:

```
<loop> ::= "FOR" "(" <integer_literal> ")" "{" <body> "}" ";"
```

The same rules for the <body> as for function bodies apply here. Having a “while” type loop was deemed unnecessary, since the parsing of the script is concluded before any feedback comes in, rendering the concept of a conditional obsolete.

Once all variables and functions have been defined, the synthesis script commences with the “MAIN” keyword:

```
<script> ::= "MAIN" "{" <body> "}" ";"
```

Everything within the main body is what actually gets executed.

The parsing of the ChASM source code is accomplished using PLY,¹³⁸ a pure Python implementation of the popular Lex and YACC utilities. During lexical analysis prior to parsing, anything between a “#” and the end of a line is ignored, allowing both in-line comments and line-spanning comments. After lexical analysis, functions and variables are resolved by simple replacements, and loops are resolved by copying the content as required. Ultimately, a list containing only instructions is compiled, which is then executed one by one by the Chempiler.

7.3.8 ADDITIONAL MODULES AND FINAL STRUCTURE

During development of the Chempiler it became increasingly clear that being able to execute ChASM files on any computer without having to connect all the hardware of the platform was highly desirable. To that end, dummy classes mirroring the device classes provided by the driver packages were implemented. Those dummy classes had identical methods, but rather than connecting to hardware, they just logged the method call and the parameters *via* the Python logging module. By introducing a “simulation” switch in the user interface, ChASM scripts could now be executed with platform graphs, but without the physical platform. The result of such a simulation run would be a file with a long list of method calls.

While the overall diagnostic value of the simulation output is debatable as the files are very large (simulating the diphenhydramine hydrochloride synthesis results in 22,545 lines of output), any problems with the ChASM, the graph, or the general synthetic operations would raise an error. Thus, a user can at least rule out failures due to a missing semicolon halfway through the synthesis. Also, changes to the Chempiler itself can be tested easily, which aids ongoing development efforts.

During the development of the diphenhydramine hydrochloride synthesis using “The Script”, the state machine structure of the experimental section allowed the user to halt the execution at certain points, for example to allow the experimenter to determine yields, or oversee critical operations. To retain this valuable capability a “BREAKPOINT” keyword was added to ChASM. A breakpoint would suspend the execution of the script until a user prompted it to continue.

For diagnostic purposes all platforms were fitted with networked surveillance cameras which could be accessed from anywhere within the university intranet. This allowed us to constantly monitor the operation. The videos were also recorded and stored as research data. This approach was well suited for general observations, but the low resolution of those cameras was found to obscure many details. Using a high-resolution USB webcam would provide sufficiently clear images but recording full HD footage at 24 frames per seconds for several days would result in unmanageably

large files. A potential solution would be to reduce the frame rate, but too low a frame rate would again mean loss of potentially important details.

Ultimately, another module for the Chempiler was implemented which records variable speed video. By adding a "SET_RECORDING_SPEED" keyword to ChASM, the user could now control the frame rate of an HD camera dynamically. Crucial operations could be recorded in real time, while non-critical operations or wait times could be sped up at will, creating a time lapse. This was implemented by using OpenCV¹³⁹ for Python to grab frames from the camera at specified intervals, and add them to a video file. Furthermore, a time stamp, the current recording speed, and the most recent debug message were added to the frames, so the video could be easily correlated to the ChASM and log file.

Using this module, videos of reasonable size and length of all three syntheses were recorded. Those videos can be obtained from the Cronin Group upon request.

During early testing an issue with the volume tracking was experienced. If the synthesis was stopped for some reason, all the information regarding the current volumes was lost. In order to resume correct operation, the original graph had to be updated manually, which was inconvenient and error-prone. Furthermore, failing to reset those volumes in the graph to the proper starting volumes before running another synthesis led to obvious inaccuracies. Thus, a crash dump was implemented which would write all current node volumes to a file after every successful operation. When resuming a synthesis, the user could opt to read from the crash dump, thereby restoring the current volumes.

The structure and operation of the current version of the Chempiler at the time of writing is depicted in Figure 70. The Chempiler client provides a rudimentary user interface. The user is asked for an experiment code, which is used to label all output, as well as paths to a graph and ChASM file. Then the script asks if a video should be recorded, if the crash dump should be loaded, and if this run is a simulation. The client then invokes the Chempiler and passes the user information.

The Chempiler loads the graph and instantiates and initialises the devices. It then parses the ChASM file into a list of commands, which are passed to the command

RESULTS AND DISCUSSION

dispatcher one by one. The dispatcher determines the target executioner and prompts it to execute the command.

The code repository contains a detailed readme file detailing how to install and use the Chempiler suite. All requirements are collated in a requirements file, so the user can conveniently install all dependencies by invoking

```
pip install -r requirements.txt
```

from the command line. The code repository including all documentation as well as all files used in the syntheses described in chapter 8 are available from the Cronin Group upon request.

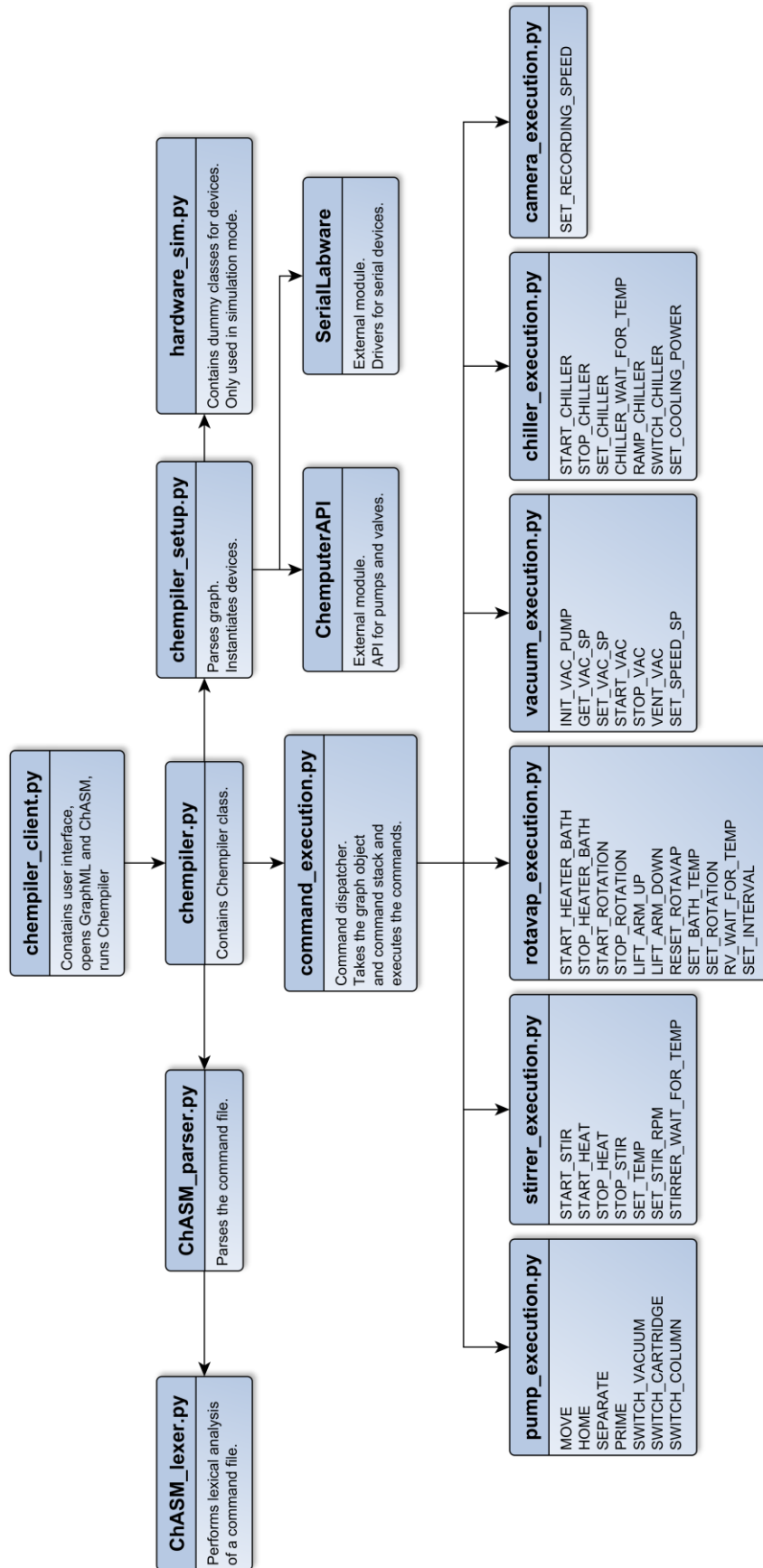


Figure 70: Module Dependency Graph of the final Chempiler suite.

7.3.9 TRANSLATION OF A SYNTHETIC PROCEDURE INTO CHASM

The process of translating a given procedure into executable ChASM code always follows the same basic workflow, demonstrated in the following example. Consider the first step of the diphenhydramine hydrochloride synthesis (chapter 8.1.4), a Grignard reaction. The starting point is a written scheme of work, as would be found in a publication. The following procedure has been adapted from Ahmadi *et al.*¹⁴⁰:

A 250 mL round bottomed flask with reflux condenser and argon inlet was charged with magnesium grit (2.50 g, 102.7 mmol) and diethyl ether (40 mL). Bromobenzene (neat, 2.95 g, 2.0 mL, 18.8 mmol) was added slowly under stirring. The mixture was heated to reflux (50°C) and stirred for 20 minutes to initiate the Grignard formation.

After cooling below 25°C the rest of the bromobenzene (12.75 g, 8.65 mL, 81.2 mmol) was added dropwise under stirring. The mixture was then stirred for 20 minutes at room temperature followed by 20 minutes at reflux (50°C).

After the Grignard formation was finished, benzaldehyde (2 M in diethyl ether, 50 mL, 0.1 mol) was added slowly. The mixture was then held at reflux and stirred for 5 hours.

After the mixture cooled below 30°C, the reaction was quenched with water (25 mL) and hydrochloric acid (2 M, 100 mL). The layers were separated and the organic layer was washed with water (50 mL), transferred to the rotary evaporator and concentrated *in vacuo* (700 mbar), yielding 15.88 g (86%) of crude diphenylmethanol as a yellow solid.

First, the required chemicals and solvents and their corresponding amounts (green highlight) as well as the physical parameters (blue highlight) are identified:

A 250 mL round bottomed flask with reflux condenser and argon inlet was charged with magnesium grit (2.50 g, 102.7 mmol) and diethyl ether (40 mL). Bromobenzene (neat, 2.95 g, 2.0 mL, 18.8 mmol) was added slowly under stirring. The mixture was heated to reflux (50°C) and stirred for 20 minutes to initiate the Grignard formation.

RESULTS AND DISCUSSION

After cooling below 25°C the rest of the bromobenzene (12.75 g, 8.65 mL, 81.2 mmol) was added dropwise under stirring. The mixture was then stirred for 20 minutes at room temperature followed by 20 minutes at reflux (50°C).

After the Grignard formation was finished, benzaldehyde (2 M in diethyl ether, 50 mL, 0.1 mol) was added slowly. The mixture was then held at reflux and stirred for 5 hours.

After the mixture cooled below 30°C, the reaction was quenched with water (25 mL) and hydrochloric acid (2 M, 100 ml). The layers were separated and the organic layer was washed with water (50 mL), transferred to the rotary evaporator and concentrated in vacuo (700 mbar), yielding 15.88 g (86%) of crude diphenylmethanol as a yellow solid.

Those parameters are then defined as variables at the beginning of the ChASM file:

```
# volumes
DEF volume_ether_grignard = 30; # mL
DEF volume_bromobenzene_1 = 2; # mL
DEF volume_bromobenzene_2 = 8.65; # mL
DEF volume_ether_bromobenzene_flush = 10; # mL
DEF volume_benzaldehyde = 50; # mL, for a 2M solution
DEF volume_water_quench = 25; # mL
DEF volume_acid_dilution = 100; # mL
DEF volume_ether_sweep_up = 5; # mL
DEF volume_water_wash = 50; # mL

# stirring rates
DEF grignard_stirring_rate = 250; # rpm

# temperatures
DEF grignard_temperature = 50; # °C

# times
DEF wait_time_grignard_initiation = 1200; # in s, actually 20min
DEF wait_time_grignard_formation = 900; # in s, actually 15min
DEF wait_time_grignard_reaction = 18000; # in s, actually 5h
DEF wait_time_grignard_extraction = 1800; # in s, actually 30min
```

The stirring rate is not explicitly stated, but usually empirical values for appropriate stirring speeds are known to the experimenter, or a common default value may be used. Note how the 40 mL of ether are split into two portions, the reason for which will become apparent a bit later on. Magnesium grit is a solid, so it will have to be charged manually. Ergo, there is no need to define a variable for it. Benzaldehyde is a liquid and could in principle be added neat, but due to the exothermic nature of the reaction, adding it as a pre-made solution is preferred. The variable `volume_ether_sweep_up` will be used for flushing to ensure quantitative transfer, much as a human chemist would do by hand.

Then, the required unit operation modules are identified. The reaction itself demands a mixing/heating module. The workup requires a liquid/liquid separation (cf. “The layers were separated...”) and an evaporation (cf. “concentrated *in vacuo*”). Those modules, as well as storage vessels for the required chemicals, are then arranged in the graph. Practically, this mostly means just distributing the reagents sensibly, as the modules are already present and connected. The rules for the graph are described in chapter 7.3.6. For drawing the graph, it is recommended to use an existing graph as template. The arrangement of the modules and chemicals in the graph is governed by a number of considerations. First, chemicals that react adversely when mixed should not share a valve due to the risk of cross-contamination. For example, it was found that having acidic and basic reagents on the same valve often leads to salt precipitation and subsequent blockage. Aqueous reagents should be separated from organic ones if dryness is a concern. In the diphenhydramine hydrochloride synthesis, a layout was devised where the right half of the platform deals with aqueous solutions, while the left half is exclusively reserved for organic materials. Appropriate washing solvents (i.e. water; a water-miscible organic solvent such as acetone, ethanol, or isopropanol; and a dry organic solvent such as diethyl ether or dichloromethane) should be placed on the outermost valves to facilitate Backbone cleaning. Last but not least, transfer lengths should be minimized, especially for small quantities.

It is imperative that nodes are given unique, expressive, and memorable names. In this work, the prefix “flask_” was used for all reagent bottles to clarify that this refers

to the node, as opposed to the chemical inside. However, this is not technically necessary.

The GraphML files used by the Chempiler have a number of additional data fields as described in chapter 7.3.6. Those must be populated with the appropriate metadata such as vessel volume, as well as with all data required to interface with attached hardware. Empty fields are disregarded by the Chempiler software.

With a graph in hand, the individual operations required to carry out the synthesis are identified. Consider the first paragraph:

A 250 mL round bottomed flask with reflux condenser and argon inlet was charged with magnesium grit (2.50 g, 102.7 mmol) and diethyl ether (40 mL). Bromobenzene (neat, 2.95 g, 2.0 mL, 18.8 mmol) was added slowly under stirring. The mixture was heated to reflux (50°C) and stirred for 20 minutes to initiate the Grignard formation.

Charging the magnesium grit has already been done by hand. This means the real first step is moving ether to the reactor. The ChASM reference (see appendix IV) specifies the `MOVE` command as follows:

```
MOVE({src}, {dest}, {volume}, {move_speed}, {aspiration_speed},  
{dispense_speed});
```

In this case `{src}` is `flask_ether`, the name for the ether bottle as defined in the graph. `{dest}` is `reactor_reactor`, again as defined in the graph. For `{volume}` the previously defined variable `volume_ether_grignard` is provided. This way, tweaking the synthesis in the future only involves changing the variable definition. The three speed values are optional as described in the ChASM reference, and default to values specified in the Chempiler constants file. However, since diethyl ether has a high vapor pressure and tends to cavitate when aspirated too quickly, the aspiration speed should be decreased. To that end, it is advisable to define a few different default pumping speeds at the top of the ChASM file. One then has to provide the `{move_speed}` and the `{aspiration_speed}` since arguments in ChASM are strictly positional. `{dispense_speed}` may be omitted

and will then default to the standard pump speed. Thus, the first line of the synthesis looks like this:

```
S MOVE(flask_ether, reactor_reactor, volume_ether_grignard, \
default_speed_fast, default_speed_slow);
```

The ChASM grammar demands that every line ends with a semicolon. The line break (signified by a backslash) in the above example was introduced to improve readability on an A4 page, in the original ChASM file, this would be one line.

Next, the recipe demands slow addition of the first portion of bromobenzene under stirring. The ChASM reference provides two commands required to stir the reactor:

```
SET_STIR_RPM({name}, {rpm});

START_STIR({name });
```

Consequently, the next two lines of the ChASM script are:

```
S SET_STIR_RPM(reactor_reactor, grignard_stirring_rate);
S START_STIR(reactor_reactor);
```

The addition of the bromobenzene works similar to the ether with the only difference that now aspiration and move speed may remain fast, but the dispense speed is set to slow:

```
S MOVE(flask_bromobenzene, reactor_reactor, volume_bromobenzene_1, \
default_speed_fast, default_speed_fast, default_speed_slow);
```

At this point it is to be reasoned that some of the bromobenzene will still be in the tube leading to the reactor, rather than in the reactor itself. To rectify this, another small portion of ether is pumped into the reactor. Similarly, if a chemist were to perform this reaction by hand, they would most likely use some solvent to flush all the bromobenzene into the reactor as well. This is the reason why the 40 mL of ether specified in the recipe were split into two smaller portions, `volume_ether_grignard` (30 mL) and `volume_ether_bromobenzene_flush` (10 mL). This addition works exactly as the first ether portion.

Now the recipe specifies refluxing the mixture for 20 minutes. Starting the heating function of the hotplate works analogous to the stirring. To make sure the mixture is refluxing before the wait time commences, the `STIRRER_WAIT_FOR_TEMP` command may be used. This command monitors the current temperature and delays execution of the script until the temperature is within a margin around the setpoint temperature. Once temperature has been reached, an additional `WAIT` command is used. Thus, the first paragraph of the recipe translates into ChASM as follows:

```
# charging reactor and initiating Grignard
S MOVE(flask_ether, reactor_reactor, volume_ether_grignard, \
default_speed_fast, default_speed_slow);
S SET_STIR_RPM(reactor_reactor, grignard_stirring_rate);
S START_STIR(reactor_reactor);
S MOVE(flask_bromobenzene, reactor_reactor, volume_bromobenzene_1, \
default_speed_fast, default_speed_fast, default_speed_slow);
S MOVE(flask_ether, reactor_reactor, volume_ether_bromobenzene_flush, \
default_speed_fast, default_speed_slow);

# stirring and waiting
S SET_TEMP(reactor_reactor, grignard_temperature);
S START_HEAT(reactor_reactor);
S STIRRER_WAIT_FOR_TEMP(reactor_reactor);
S WAIT(wait_time_grignard_initiation);
```

The lines beginning with the `#` symbol are comments and will be ignored by the Chempiler parser. It is advisable to add comments like those describing what operation a code block accomplishes. This makes it a lot easier to read and understand a script, and to track down problems.

The next paragraph works much the same as the previous one:

After cooling below 25°C the rest of the bromobenzene (12.75 g, 8.65 mL, 81.2 mmol) was added dropwise under stirring. The mixture was then stirred for 20 minutes at room temperature followed by 20 minutes at reflux (50°C).

After the Grignard formation was finished, benzaldehyde (2 M in diethyl ether, 50 mL, 0.1 mol) was added slowly. The mixture was then held at reflux and stirred for 5 hours.

RESULTS AND DISCUSSION

Again, the operations are identified one by one and translated into ChASM commands as demonstrated above. The only notable difference is that after addition of the reagents, it is advisable to wash the Backbone, to avoid cross-contamination or blockage. This is accomplished by repeatedly pumping ether across the entire Backbone. For repeated operations, the ChASM language provides a primitive FOR loop syntax. Pumping ether across the Backbone three times is written as:

```
FOR(3){  
  S MOVE(flask_ether, waste_reactor, 5, default_speed_fast, \  
  default_speed_slow);  
}
```

Note the use of the number “5” to specify the volume. This is called a “literal”. Ideally, this should be defined as a variable as well to improve code maintainability, but in this case, it serves as an example. Also note how the Backbone is only cleaned up until the reactor. The valves to the right of the reactor have not yet been used in the synthesis and are assumed to be clean, so in order to save time, only the contaminated section is cleaned.

After having cleaned the Backbone with ether, it is advisable to empty the lines again by pumping air through. This reduces cross-contamination and material exposure to solvents. Air may be obtained from the waste outlet. The waste tubes should never reach to the bottom of the waste container, thus allowing the user to draw air through them. In this case, flushing with air means pumping a volume from the rightmost valve back to the leftmost valve:

```
S MOVE(waste_reactor, waste_reagents_1, 5); # flush lines with air
```

For cooling to room temperature, the STIRRER_WAIT_FOR_TEMP command is used again. Since that command always waits until the current setpoint is reached, the setpoint first has to be set to room temperature:

```
# cool to room temperature  
S STOP_HEAT(reactor_reactor);  
S SET_TEMP(reactor_reactor, room_temperature);  
S STIRRER_WAIT_FOR_TEMP(reactor_reactor);
```

RESULTS AND DISCUSSION

Where `room_temperature` is a variable defined at the top of the script. Other than that, the same considerations as discussed for the first paragraph apply, yielding the following code for the second and third paragraph:

```
# cool to room temperature
S STOP_HEAT(reactor_reactor);
S SET_TEMP(reactor_reactor, room_temperature);
S STIRRER_WAIT_FOR_TEMP(reactor_reactor);

# add rest of the bromobenzene
S MOVE(flask_bromobenzene, reactor_reactor, volume_bromobenzene_2, \
default_speed_fast, default_speed_slow, 1); # dispense dead slow
S MOVE(flask_ether, reactor_reactor, volume_ether_bromobenzene_flush, \
default_speed_fast, default_speed_slow);

# wash Backbone
FOR(3){
    S MOVE(flask_ether, waste_reactor, 5, default_speed_fast, \
default_speed_slow);
}
S MOVE(waste_reactor, waste_reagents_1, 5); # flush lines with air

# wait for formation to complete
S WAIT(wait_time_grignard_formation);

# heat to reflux and wait again
S SET_TEMP(reactor_reactor, grignard_temperature);
S START_HEAT(reactor_reactor);
S WAIT(wait_time_grignard_formation);

# add the benzaldehyde
S SET_STIR_RPM(reactor_reactor, 250);
S MOVE(flask_benzaldehyde, reactor_reactor, volume_benzaldehyde, \
default_speed_fast, default_speed_slow, 1); # dispense dead slow

# wash Backbone
FOR(3){
    S MOVE(flask_ether, waste_reactor, 5, default_speed_fast, \
default_speed_slow);
}
S MOVE(waste_reactor, waste_reagents_1, 5); # flush lines with air

# waiting for reaction completion
S WAIT(wait_time_grignard_reaction);
```

RESULTS AND DISCUSSION

The last paragraph of the recipe describes the quench and isolation, a straightforward aqueous workup followed by solvent evaporation:

After the mixture cooled below 30°C, the reaction was quenched with water (25 mL) and hydrochloric acid (2 M, 100 ml). The layers were separated and the organic layer was washed with water (50 mL), transferred to the rotary evaporator and concentrated *in vacuo* (700 mbar), yielding 15.88 g (86%) of crude diphenylmethanol as a yellow solid.

The quench consists of moving water and hydrochloric acid to the reactor as has been described already. Additionally, to ensure an efficient quench, the stirring rate is increased. Also, it takes a while for the magnesium salt to dissolve, so an additional waiting period is introduced before transferring the mixture to the separator. Those two pieces of information are not mentioned in the recipe, which is a good example of hidden knowledge required to successfully replicate a synthetic procedure. A human chemist might increase the stirring rate regardless of the recipe, as well as recognize the sluggish dissolution and compensate for it. Those observations are sometimes captured in the lab notebook, but they rarely are mentioned in published procedures. Scripting syntheses in ChASM alleviates this problem, as every step is necessarily described in full detail.

The transfer to the separator is also worth mentioning. After transferring the bulk of the mixture, there are in all likelihood still small amounts of product remaining in the flask. A human chemist would flush them over using small amounts of solvent. Thus, in the automated procedure, small amounts of ether and hydrochloric acid are added to the reactor, the reactor is stirred briefly, and the mixture is transferred to the separator. This is repeated three times using a `FOR` loop. Again, this is not explicitly mentioned in the recipe, but any practicing synthetic chemist would attempt to flush out all residues of product in some fashion. This goes to show that a certain level of laboratory experience and attention to detail is necessary to successfully encode a synthesis.

Taking into account all those hidden steps, the complete code for the quench would be:

RESULTS AND DISCUSSION

```
# cooling down
S STOP_HEAT(reactor_reactor);
S SET_TEMP(reactor_reactor, room_temperature);
S STIRRER_WAIT_FOR_TEMP(reactor_reactor);

# aqueous workup
S SET_STIR_RPM(reactor_reactor, extraction_stirring_rate);
S MOVE(flask_water, reactor_reactor, volume_water_quench);
S WAIT(60);
S MOVE(flask_HCl, reactor_reactor, volume_acid_dilution);
S WAIT(wait_time_grignard_extraction);
S STOP_STIR(reactor_reactor);
S WAIT(60);

# move to separator
S MOVE(reactor_reactor, flask_separator_top, all, default_speed_fast, \
default_speed_slow); # aspirate more slowly
FOR(3){
  S MOVE(flask_HCl, reactor_reactor, 5);
  S MOVE(flask_ether, reactor_reactor, 5, default_speed_fast, \
default_speed_slow);
  S START_STIR(reactor_reactor);
  S WAIT(60);
  S STOP_STIR(reactor_reactor);
  S MOVE(reactor_reactor, flask_separator_top, all, \
default_speed_fast, default_speed_slow);
}
```

Note the use of the `all` keyword instead of a volume in the line that moves the contents of the reactor to the separator. `all` takes advantage of the internal volume tracker and prompts the Chempiler to move however much volume currently occupies a given vessel. This is a handy feature every time the entire contents of a vessel should go somewhere.

Next, the procedure simply states, “The layers were separated”. A human chemist would probably shake the separating funnel before separating the layers to ensure an efficient extraction. To replicate this good practice, it is advisable to stir the separator. Although the separator uses an overhead stirrer instead of the hotplate stirrer of the reactor, due to the way the Chempiler handles hardware modules, the same commands to set the stirring speed and start and stop the stirrer apply:

RESULTS AND DISCUSSION

```
S SET_STIR_RPM(flask_separator_top, extraction_overhead_stirring_rate);  
S START_STIR(flask_separator_top);  
S WAIT(60);
```

A common laboratory trick to improve phase separation in a separatory funnel is to gently swirl it, in order to detach droplets clinging to the walls, and encouraging foamy interphase layers to collapse. This strategy can be replicated by gently stirring the separator at a very low RPM:

```
S SET_STIR_RPM(flask_separator_top, extraction_settling_stirring_rate);  
S WAIT(wait_time_settling);  
S STOP_STIR(flask_separator_top);
```

The actual phase separation is handled below the user level; therefore, it is a single command:

```
SEPARATE({lower_phase_target}, {upper_phase_target})
```

Since the separation process is a slow one, it is advisable to move the majority of the lower layer out first, based on a rough estimation of the amounts. This speeds up the whole operation considerably. Thus, the code for the separation is as follows:

```
# do the separation, keep the organic layer in the separatory funnel  
S MOVE(flask_separator_bottom, waste_workup, 100);  
S SEPARATE(waste_workup, flask_separator_top);
```

Next, the recipe calls for the organic layer to be washed with water. This consists of the operations of moving water into the separator, stirring the mixture, letting it settle, disposing of the bulk of the aqueous layer, and running a separation; all of which have been discussed before. It has been found to be advantageous to add aqueous washes through the bottom of the separator rather than the top, to avoid trapping organic solvent in the outlet, which might impede sensor readings, or decrease yield.

```
# add water  
S MOVE(flask_water, flask_separator_bottom, volume_water_wash);  
  
# vigorous stirring, then settling
```

RESULTS AND DISCUSSION

```
S SET_STIR_RPM(flask_separator_top, extraction_overhead_stirring_rate);
S START_STIR(flask_separator_top);
S WAIT(60);
S SET_STIR_RPM(flask_separator_top, extraction_settling_stirring_rate);
S WAIT(wait_time_settling);
S STOP_STIR(flask_separator_top);

# do the separation, transfer the top layer to the rotavap
S MOVE(flask_separator_bottom, waste_workup, 40);
S SEPARATE(waste_workup, rotavap);
```

Note how this time the upper phase target is the rotary evaporator, as the next step is evaporation to dryness.

As discussed for the transfer from reactor to separator, there will be some material left in the separator after this operation. A human chemist would undoubtedly flush the remainder out of the separating funnel with small portions of solvent, so the same approach should be taken here. Using a `FOR` loop, ether is moved to the separator, stirred, and then moved to the rotary evaporator:

```
# sweeping up
FOR(3) {
    S MOVE(flask_ether, flask_separator_top, 5, default_speed_fast, \
    default_speed_slow);
    S SET_STIR_RPM(flask_separator_top, 1000);
    S START_STIR(flask_separator_top);
    S WAIT(60);
    S STOP_STIR(flask_separator_top);
    S MOVE(flask_separator_bottom, rotavap, all, default_speed_fast, \
    default_speed_slow);
}
```

Having successfully transferred the organic layer to the rotary evaporator, the ether now needs to be removed. In the course of this work, a library function `evaporate_to_dryness` was developed which takes care of that. A detailed description of the process and the rationale behind it can be found in chapter 6.2.2, but here this function can be used as a black box. As of yet, ChASM does not support import of other files, so the function definition has to exist in the same file. However,

it may just be copy-pasted from an existing synthesis. `evaporate_to_dryness` takes the following arguments, in that order:

- Distillate volume
- End vacuum
- Vacuum pump speed
- Distillation time
- Drying time (at max. vacuum)

For the time being, those parameters have to be found empirically. In future iterations of the platform, improved instrumentation may enable the Chempiler to evaporate solvent autonomously, but for now a certain level of guesswork is required. For this synthesis, the following parameter set was found to be suitable:

```
# rotating off solvent
evaporate_to_dryness(115, 700, 100, 1800, 3600);
```

After the product is obtained successfully, one last task has to be taken into consideration, that is cleaning of the used glassware in order to prime it for the next use. Due to the nature of the Grignard chemistry, in this case it is advisable to first clean both vessels with dilute HCl to remove residual magnesium salt. This is achieved by moving hydrochloric acid and water into the reactor, stirring, then moving the mixture into the separator, stirring again, and then moving the contents to waste. Once the magnesium salt is taken care of, the library functions `clean_reactor` and `clean_separator` may be used to properly clean out the vessels:

```
# cleaning out the remaining salt
S MOVE(flask_HCl, reactor_reactor, 25);
S MOVE(flask_water, reactor_reactor, 75);
S START_STIR(reactor_reactor);
S WAIT(900); # wait for 15min for all solid to dissolve
S STOP_STIR(reactor_reactor);
S MOVE(reactor_reactor, flask_separator_top, all);

S SET_STIR_RPM(flask_separator_top, 1000);
S START_STIR(flask_separator_top);
S WAIT(60);
S STOP_STIR(flask_separator_top);
```

```
S MOVE(flask_separator_bottom, waste_workup, all);
```

```
clean_reactor();
```

```
clean_separator();
```

Ultimately, the code obtained has to be validated experimentally, and sometimes requires optimization. The ChASM file used in the successful diphenhydramine hydrochloride syntheses for example was slightly different from what is described in here, for various practical reasons. Nevertheless, the methods described above provide a good starting point. It was found to be helpful to picture the steps one would take in the laboratory if one would manually reproduce a synthesis. Breaking down those manual operations into their fundamental constituents and typing them up using the ChASM notation is a very natural way of digitising a given synthesis.

Once a ChASM file is successfully scripted, the Chempiler offers a simulation mode (see chapter 7.3.8) which allows the user to run the ChASM on a platform graph and see if any errors occur. After successfully simulating the synthesis, workflow was adopted whereby the code would first be executed using an empty platform without chemicals or solvents. This was called a “dry run”, the purpose of which is to verify correct operation without wasting chemicals and risking potential adverse reactions. When satisfied with the dry run, one would then normally follow up with a “breakpoint run” where breakpoints (see chapter 7.3.8) are inserted at neuralgic points in the sequence. This would allow observation of critical operations, sampling, determination of yields and purities of intermediates and so forth. Once the breakpoint run completed without problems, the breakpoints were removed, and the entire synthesis was executed in one automated full run.

RESULTS AND DISCUSSION

8 AUTOMATED SYNTHESSES

Three targets were chosen to demonstrate key capabilities of the synthesiser (Figure 71): diphenhydramine hydrochloride (**1**), rufinamide (**2**), and sildenafil (**3**). Diphenhydramine hydrochloride (**1**) is an ethanolamine derivative used as antihistamine and mild sleep aid and is marketed as NytolTM in the UK and Benadryl[®] in the US. The synthesis is a four-step sequence starting with a Grignard reaction, which was deemed a worthwhile challenge to automate. Rufinamide (**2**) is a triazole derivative used as an anticonvulsant to treat various seizure disorders. It is marketed as Inovelon[®] in the EU and as Banzel[®] in the US. The synthesis is a three step one pot procedure featuring a triazole click reaction. It was chosen it for its relatively simple synthesis, to demonstrate the capabilities of a minimal setup, as well as the interoperability between two different platforms arising from the Chempiler concept (see chapter 7.3). To that end, the same ChASM file was successfully executed on two physically different platforms, obtaining comparable yields and purities. Sildenafil (**3**) is prescribed to treat erectile dysfunction and is best known under the brand name VIAGRA[®]. The industrial synthesis route¹⁴¹ features a chlorosulfonation with highly aggressive chlorosulfonic acid and thionyl chloride. In the author's view this step both demonstrates the robustness of the platform and highlights the safety benefit arising from automating dangerous procedures.

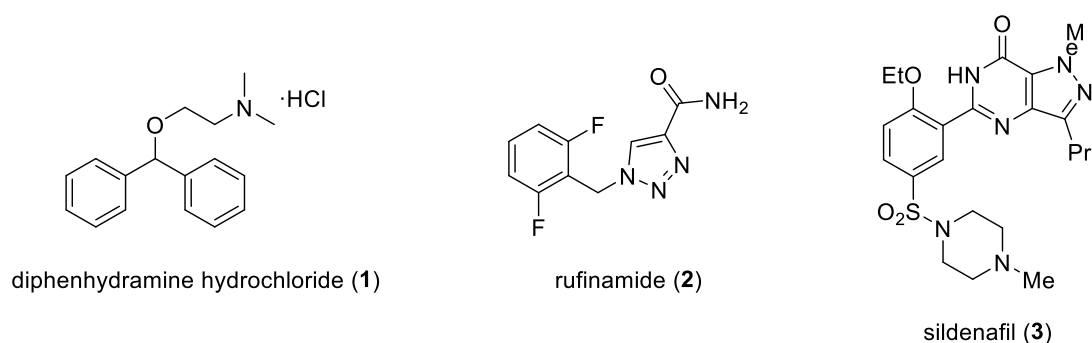


Figure 71: The three target molecules.

8.1 SYNTHESIS OF DIPHENHYDRAMINE HYDROCHLORIDE (1)

8.1.1 PREREQUISITES AND INITIAL WORK

The antihistamine drug diphenhydramine was first discovered by Rieveschl and Woods in 1944¹⁴² and various derivatives have been shown to possess similar activity.¹⁴³ Figure 72 shows one synthetic route adapted from Ahmadi *et al.*¹⁴⁰ Automating the initial Grignard step was especially intriguing, since it is not a trivial operation. Grignard reactions require dry conditions, initiating them can be a challenge, and the exothermic nature makes them challenging from a safety standpoint. Thus, an automated Grignard reaction would constitute a powerful display of the system's capabilities.

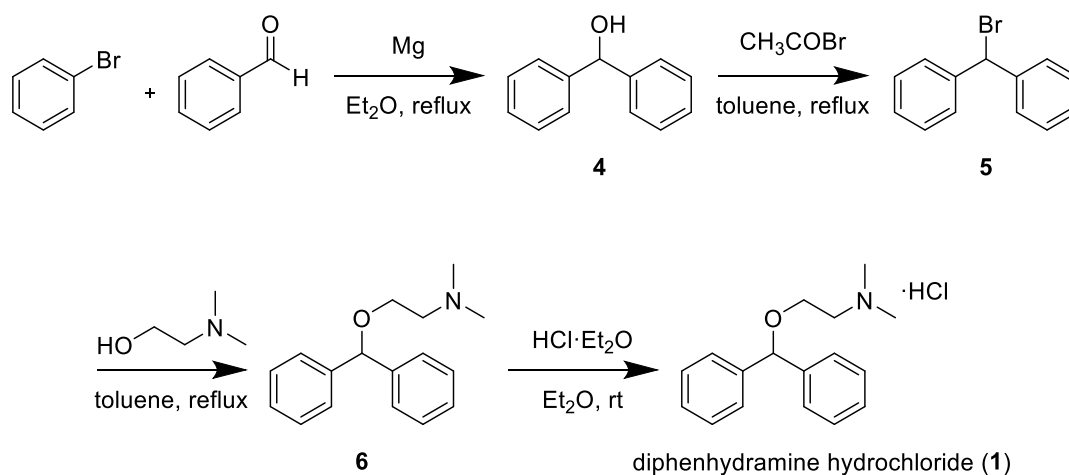


Figure 72: Synthetic sequence for diphenhydramine hydrochloride, adapted from Ahmadi *et al.*¹⁴⁰ The original procedure reported in the paper called for benzene as the solvent for the second step, and xylene for the third.

A colleague successfully replicated the synthesis manually, thereby producing valuable information and experience for me to build on. Additionally, all intermediates except for the diphenhydramine free base were commercially available, so the automation of the sequence could commence at any point and every step could be optimised in isolation before tying them all together into one automated multistep synthesis.

Based on the published synthetic procedure, all required reagents and solvents were identified, as well as the required unit operations of mixing/refluxing, liquid/liquid

separation, evaporation, and filtration. Subsequently a platform initially consisting of five Backbone units, a reflux reactor (see chapter 6.2.1), and the automated rotary evaporator (see chapter 6.2.2) was built.

8.1.2 THE JOURNEY BEGINS WITH A BROMINATION

Of all steps in the sequence, the transformation of the diphenylmethanol **4** to bromodiphenylmethane **5** was the simplest in terms of unit operations. The initial recipe called for a benzene solution of **4** to be added to neat acetyl bromide, refluxed for 11 h, and evaporated to dryness to yield **5** as brown oil. As moving material around the platform was a solved problem (see chapter 7.3.5), the main development required for this step was the automated evaporation.

At this point all software control was achieved through one monolithic Python script containing both the various required functions, and the actual synthesis code. This was always meant to be a temporary hack and it was meant to be replaced by what would later become the Chempiler (see chapter 7.3). However, due to repeated delays in the development of the control suite this single Python script was used for most of the development work on the diphenhydramine synthesis and would eventually grow to over 2,000 lines. Nevertheless, it allowed me to run the synthesis nearly end-to-end, and in the process to learn how to do chemistry on the platform.

The initial idea was to manually charge diphenylmethanol into the rotavap flask to simulate a successful first step. Then, the system was to pump acetyl bromide into the reactor, and automatically dissolve and transfer the diphenylmethanol to the reactor as well. To that end, three portions of xylene were pumped into the rotavap, and the flask was rotated in interval mode to facilitate dissolution. This rather unusual feature of the IKA rotary evaporator allows the user to specify an interval in seconds, after which the rotation stops and reverses direction. After another interval has passed, the rotation stops and reverses again, and so forth. While clearly not useful for evaporations, for tasks such as redissolving solids or drying powders this interval mode was found to perform better than continuous rotation.

RESULTS AND DISCUSSION

After successfully formulating the reaction mixture, the heating and stirring of the hotplate stirrer was switched on, and the mixture was held at reflux overnight. To subsequently evaporate the solvent, the plan was to transfer the mixture to the rotary evaporator, lower the flask into the heating bath, and distil off the bulk of the solvent. Once the distillation is finished, the rotavap could be vented and the distillate could be transferred to waste. This was then to be followed by drying of the residue at full vacuum.

After adjusting the volumes and dissolution times a bit, the first operations worked surprisingly well. One finding was that the hypodermic needle used in the reactor was corroded heavily by the aggressive acetyl bromide. This was corroborated by broad line shapes due to poor shimming in the NMR of the product, suggesting the presence of paramagnetic iron salts. The needle was subsequently replaced with a PTFE tube directly inserted into the reactor, as described in detail in chapter 6.2.1. Since the reflux condenser was open to the atmosphere, a significant amount of acetyl bromide, or decomposition products thereof, escaped into the fume hood, heavily corroding metal parts in the vicinity. The reaction mixture was also dark brown, almost black. All this would later be addressed by the addition of an inert gas system (see chapter 6.2.5). Aside from those minor issues, the reaction itself proceeded uneventfully.

The evaporation, however, posed a bigger challenge. First of all, a major flaw in the design was revealed when upon venting, the oily residue was forced back into the inlet tube, subsequently crystallising and blocking the tube. This problem was easily solved by addition of a magnetic lifting system as described in chapter 6.2.2.

A more persistent issue was experienced with regards to the evaporation itself. The vacuum pump employed, a Vacuubrand MD1C Vario plus (Figure 73) provides a so-called "Auto mode", which supposedly finds the boiling point of a solvent or mixture automatically and adjusts the vacuum to provide optimal distillation.



Figure 73: Vacuubrand MD1C Vario plus vacuum pump. Image reproduced with permission from Asynt Ltd.

Unfortunately, in our hands this feature never worked properly. Manual investigations revealed that at 40°C, the benzene distilled over satisfactorily at around 150 mbar. However, automatic boiling point determination would identify distillation at pressures as high as 700 mbar. Furthermore, the pressure would not remain at that level but rather be slowly reduced over the course of ten to fifteen minutes to below 100 mbar, at which point the mixture would boil and foam violently and bump uncontrollably.

After spending significant time tweaking parameters, corresponding with the pump manufacturer, and experimenting with different solvents, this feature still could not be utilised properly, so it was ultimately abandoned for a hard-coded approach. Henceforth, pressures and times for every evaporation were determined experimentally prior to automation, and then passed to the evaporation function as variables. Thus, the following procedure was established for evaporating a product solution to dryness.

Solvent evaporation started with pumping the solution to be evaporated into the distillation flask of the rotary evaporator. The flask was then lowered into the heating bath, and the rotation was started. The vacuum pump was started, lowering the

RESULTS AND DISCUSSION

pressure to 900 mbar in order to degas the solution and avoid excessive foaming later on. The heating bath and the recirculation chiller servicing the condenser were switched on, the target temperatures were set, and execution of the script was suspended until the target temperatures were reached. The vacuum setpoint was then changed to the target distillation pressure, and the vacuum pump speed was adjusted according to the solvent, in order to avoid bumping. Then, the execution of the script was suspended for a user-defined amount of time to allow the main distillation to finish.

After the distillation was complete, the flask was lifted up which caused the inlet tube to be attracted by the magnet (Figure 42), lifting it out of the remaining solution. The vacuum pump was subsequently stopped, and the vacuum was vented. Then, a user-defined amount of distillate was removed from the distillate flask and discarded.

At this point, it was found that proceeding directly to drying the product under maximal vacuum would often lead to a few millilitres of residual solvent distilling over, which decreased the drying efficiency. Thus, the flask was lowered back into the bath and the vacuum pump was set to maximum power for two minutes, drawing out any residual solvent. The sequence of raising the flask, venting the vacuum, and emptying the distillate flask was then repeated.

Next, the flask was lowered once again, the vacuum pump was set to maximum power and started, and the cooling of the condenser was switched off. The product was then dried for a user-defined amount of time.

After the drying was complete, the flask was lifted up once more, the vacuum was vented, and the rotation and heating bath were switched off.

Having successfully mastered the solvent evaporation, crude bromodiphenylmethane was obtained as dark brown oil in quantitative yields. The product still contained small quantities of acetic acid, but it was later found that this did not interfere with the subsequent Williamson ether synthesis, so no further purification was attempted.

8.1.3 PRESSING ON TOWARDS THE WILLIAMSON ETHER SYNTHESIS

Emboldened by the successful automation of the bromination step, the subsequent Williamson ether synthesis was tackled next. The main reasoning behind this decision was that, in addition to the already established unit operations of mixing/refluxing and solvent evaporation, the Williamson step demands extensive aqueous workup, thereby giving us an opportunity to study liquid/liquid separations.

The description of the workup procedure provided by Ahmadi *et al.*¹⁴⁰ was somewhat cryptic to us:

“

The reaction mixture was cooled, treated with water before the aqueous layer was extracted with ether, re-extracted with 10% HCl, neutralized with 10% NaOH, dried over MgSO₄, and evaporated under vacuum to obtain the desired oily compound.

“

It is not entirely clear to us whether this is just poorly phrased, or whether the procedure was fundamentally ill-conceived. A colleague tested several interpretations of those instructions, but always found that the product would be distributed between the various layers, and therefore could not achieve quantitative recovery of reasonably pure product. Thus, a work-up procedure was designed independent of the literature description. As the product is a basic amine, it should be extracted with aqueous hydrochloric acid, forming the water-soluble hydrochloride, and leaving behind organic impurities. This was to be followed by neutralisation with aqueous sodium hydroxide, liberating the free base, which could then be extracted with diethyl ether, leaving behind water-soluble impurities. Arguably, the literature procedure might very well be describing this very procedure, just very poorly worded.

The initial operations for formulating the reaction mixture proceeded uneventfully. Again, the rotavap flask was manually charged with pure starting material to simulate a successful previous step, and the reactor was charged with

RESULTS AND DISCUSSION

N,N-dimethylaminoethanol. It was found that the high viscosity of the neat dimethylaminoethanol required a decreased pump speed, which was subsequently implemented into the transfer routine. The bromodiphenylmethane was subsequently transferred to the reactor as described above, and the mixture was refluxed.

For the liquid/liquid separation though, problem after problem was encountered. First, the reaction itself creates a highly viscous, sticky second layer. Later considerations suggested this to be diphenhydramine hydrobromide. When the alkyl bromide reacts with the alcohol, HBr is liberated, which in turn is quenched by the highly basic amino group. The resulting salt is insoluble in organic solvents, but oddly it is also poorly soluble in water. Primarily, it clogged the tubing, making it nearly impossible to transfer any product out of the reactor. This issue was addressed by pumping the first portion of aqueous hydrochloric acid directly into the reactor, although this often led to the diphenhydramine hydrochloride and/or hydrobromide crystallising. Ultimately, it was found that when the reaction was held at 30°C during quenching rather than being cooled to room temperature, the salt would dissolve completely.

The actual separation of the organic and the aqueous layer was initially attempted by utilising a Universal Phase Separator by Biotage (Figure 44). A detailed description of the setup for this separation can be found in chapter 6.2.3. However, the Universal Phase Separator was slow and unreliable. Crucially, it often admitted copious amounts of aqueous layer to leak through. At the time, this was rectified simply by transferring leaked aqueous layer back into the separator with a pipette. As it became clear that an entirely new solution for the liquid/liquid separation had to be developed anyway, this was accepted as a temporary workaround to allow us to carry on with the chemistry development.

The evaporation was attempted in automated mode at first, but after multiple failures, the same hard-coded approach that was successful for the bromination was adopted here as well.

So far, the Williamson step was a mixed success. While the chemistry worked fine, and the new workup procedure seemed to work as well, the crucial liquid/liquid separation could not be achieved. Therefore, the focus was shifted to computer vision based solutions for the phase separation (see chapter 6.2.3 for a detailed discussion). A colleague volunteered to work on it, so it was decided to let the Williamson rest for now and instead push on to the most ambitious reaction so far, the Grignard reaction.

8.1.4 CRESTING THE FIRST SUMMIT WITH THE GRIGNARD REACTION

Grignard reactions are infamous for being fickle beasts. The reason is that starting the reaction can be tricky as metallic magnesium quickly forms an inert oxide layer, and small quantities of moisture or oxygen can quench the reaction before it takes off. Once started, however, Grignard reactions (both the formation of the reagent, and the subsequent coupling) tend to be strongly exothermic.¹⁴⁴ Combined, those two properties can easily lead to a thermal runaway.¹⁴⁵ A well-meaning experimenter may add too much of the organohalide to the magnesium metal because the reaction won't start, but then it takes off and reacts violently. As the Grignard formation requires an etheric solvent¹⁴⁶ and the most popular reaction solvent in the lab is diethyl ether, this can represent a substantial fire hazard. Therefore, on a plant scale, THF is usually favoured.

There are a number of well-established ways to activate the magnesium metal to facilitate a smooth initiation, such as iodine,¹⁴⁷ 1,2-dibromoethane,¹⁴⁶⁻¹⁴⁷ DIBAH,¹⁴⁶ crushing the magnesium,¹⁴⁸ or sonication.¹⁴⁹ The latter was immediately discarded as the required instrumentation would be unwieldy. As the Grignard formation constitutes the first step of the sequence, and solid handling was not implemented in the platform, the magnesium had to be charged by hand anyway, so it was decided to first test a combination of mechanically tearing and scraping the magnesium, and iodine. Both operations can be performed offline before the automated sequence starts, thus avoiding the need for yet another reagent input. However, initial tests were not successful. The reaction would initiate on some occasions, but the reproducibility was low, and without any process analytical techniques implemented

RESULTS AND DISCUSSION

in the platform, the abovementioned thermal runaway scenario played out on several occasions.

Eventually, the laborious and not very successful iodine activation was abandoned altogether and magnesium ribbon was replaced with magnesium grit. The grit was heated to 150°C under slow stirring and a gentle stream of argon, followed by cooling to room temperature under slow stirring. The heating and the argon flow dried the magnesium and the reaction vessel, while the stirring caused the magnesium particles to grind against each other, thereby scratching the surface oxide layer. This simple procedure could be done with little effort in parallel while preparing the other reagents, and reliably activated the magnesium.

To ensure initiation at the beginning of the automated sequence, a small portion of bromobenzene was added and the mixture was heated to reflux. After cooling to room temperature, the Grignard formation had started every time, so the rest of the bromobenzene could be added slowly at a rate that maintained a gentle reflux. Here, another advantage of the automated platform became apparent. Unlike with dropping funnels, addition speeds can be precisely controlled and accurately reproduced, so maintaining control of highly exothermic reactions is significantly easier.

After successful formation of the Grignard reagent, benzaldehyde solution in diethyl ether was added slowly, again at a rate maintaining gentle reflux. Subsequently, the mixture was refluxed for another five hours to ensure complete conversion. A large amount of precipitate was formed upon addition of the aldehyde which made the solution very hard to pump. The initial plan was to transfer the reaction mixture into an excess of saturated aqueous ammonium chloride in order to quench it safely, but this was found to be impossible. Instead, the ammonium chloride solution had to be slowly added to the reaction mixture. The exothermic quench heated the diethyl ether to reflux once more, but no detrimental effect on yield or purity was observed.

Since no robust automated phase separation was available at this point, a semi-automated approach was used instead. Ongoing investigations of both computer vision based and sensor-based systems (see chapter 6.2.3 for details) utilised an

algorithm by which a small amount of the lower phase was transferred to a target vessel. Then either a camera or a sensor would determine whether the phase boundary had been crossed. If so, the remainder of the content would be transferred to the appropriate target vessel, if not, the cycle would be repeated. To mimic this operation, a function was added to the control script that would withdraw a small portion and record a frame from the camera which was later used to train the classifier model (see chapter 6.2.3). Then, the experimenter had to press a key on the keyboard to indicate whether the phase boundary was reached or not. Once the user indicated that the boundary was reached, the script would continue the process under full automation. This approach allowed us to carry on with the chemistry development, and also generated a large number of pictures which could be used in the ongoing development of a computer vision based separator.

Using this semi-automated approach, crude diphenylmethanol was recovered in excellent yield as yellowish crystalline solid (Figure 74). The colour indicated some impurities, but the NMR showed only traces of benzaldehyde and very minor impurities in the aromatic region. Thus, further purification was not attempted.



Figure 74: Crude diphenylmethanol **4**.

The liquid/liquid separation was still an unsolved problem, but other than that full automation of a Grignard reaction was achieved, which is no small feat. Confident that the liquid/liquid separation would be solved soon, the last remaining step in the sequence was commenced.

8.1.5 COMMENCING THE END GAME WITH THE HYDROCHLORIDE PRECIPITATION

The filtration module was an unsolved problem at this point as well. Previous instances of the platform had used a relatively impractical contraption consisting of a flask, a filter, and some stainless-steel hypodermic needles (see chapter 6.2.4 for a detailed description). A number of potential solutions were proposed, and the modified Büchner funnel (Figure 50, left) was chosen as the most feasible approach. The two inputs this module required were not factored into the initial count, so the Backbone had to be extended by one unit, to a total of six Backbone units.

In the manual pre-experiments the hydrochloride precipitation posed no major challenge. Different solvents were investigated, and in methanol the recovery was rather low, but in diethyl ether the procedure worked fine.

Thus, the rotavap flask was charged with crude diphenhydramine free base obtained in earlier experiments. The dead volume of the filter was filled with diethyl ether in order to avoid hydrochloric acid leaking through the sinter plate, which could lead to precipitation inside the frit and subsequent blockage. The filter was then charged with the appropriate amount of etheric hydrochloric acid. The free base was dissolved in diethyl ether and slowly dripped into the hydrochloric acid. Initially, precipitation was sluggish, but upon prolonged stirring, a beige solid started to precipitate. Subsequently, the supernatant solution was removed through the bottom port, yielding the title compound as a moist, beige filter cake. After manual drying *in vacuo*, NMR revealed only minor impurities in the crude product. A recrystallisation was not feasible at that point due to limitations of the equipment, yet being able to obtain crude product after four steps without human intervention followed by manual recrystallisation was still seen as preferable to manually executing the entire synthesis.

8.1.6 TYING IT ALL TOGETHER

Having more or less successfully automated all four steps in isolation, it was now time to launch an attempt at running the entire sequence end-to-end. Still no workable solution for the liquid/liquid separation was in place, so the first attempt to run a full sequence utilised the previously tested semi-automatic approach.

During this first attempt, a number of things became clear. First of all, as the batch sizes of all reactions had to be coordinated, it turned out that the Grignard reaction required a 250 mL flask rather than the 100 mL flask used in the other steps. This meant that for the subsequent reactions the flask was overdimensioned. Initially this problem was circumvented by manually swapping the 250 mL flask for a 100 mL flask after the first step. The next lesson learned was that proper cleaning of the system in between operations was of paramount importance. Previously, little attention was

given to cleaning as the experiments always started from a clean rig, but for the entire four-step sequence this *laissez-faire* approach frequently led to blockages of tubing.

It also became clear that a solvent which was miscible with both water and ether was missing. Previously, lines were first flushed with water and then with ether, but this approach only displaced the water mechanically, and droplets would remain in the Backbone, interfering with water-sensitive reagents. Thus, a solvent such as acetone or isopropanol was required to flush the lines between water and ether. Alas, no free input was available anymore. Rather than adding yet another Backbone unit, it was instead decided to perform the bromination step in xylene as well. At this point there was no chemical reason for using benzene, therefore replacing it with xylene not only freed up a port on the Backbone, it also alleviated concerns regarding the toxicity of benzene.

Establishing a strict regime of cleaning the Backbone after handling reagents or reaction mixtures greatly improved the resilience of the system. However, cleaning of the modules themselves was still challenging, because the only feasible way of cleaning them was to fill them with solvent and drain it again. With the 10 mL syringes employed at the time, and the rather inefficient transfer routines (see chapter 7.3.5), this took a lot of time. Therefore, the modules were manually cleaned when they were not in use.

Enhanced understanding of the platform's behaviour led to further experimentation with various sensors for the liquid/liquid extraction. At that point, the view on computer vision was dim, so sensor-based approaches were investigated instead. Chapter 6.2.3 gives a detailed account of the evaluated sensors, until eventually a conductivity sensor was chosen. While the early prototype employed in those first full runs was crude and not very reliable, it worked well enough to suggest a properly built conductivity sensor would be a robust solution for this recalcitrant problem.

While those problems related to the chemistry and the operations could be overcome, the lack of reliability of the pumps and valves was a severe struggle. At that point, hardware with encased Hall effect sensors (see chapter 6.1.4 for a detailed

explanation) and running the old firmware (see chapter 7.1) was employed. Therefore, the platform frequently suffered Hall sensor malfunctions and device disconnects. Furthermore, when the devices disconnected, the only way to recover them was to power cycle them, so recovery from this type of malfunction by improving the control software was not possible. Additionally, the pumps were employing the NEMA23 motor (see chapter 6.1.5 for a discussion of the motor problems) and almost a dozen syringes were lost to overpressure events.

Ultimately, the author found himself with a working procedure, but hardware incapable of carrying it out. Additionally, the preliminary control script had grown to a point where maintaining it became incredibly hard, yet the ChemOS control suite (see chapter 7.3.2) was still not ready. Therefore, the chemical development was paused and the focus was shifted to fixing all the hardware and software problems. The pump and valve hardware was reworked (see chapter 6.1), the entire firmware of the devices was completely re-written (see chapter 7.1), and in collaboration with colleagues the ChemOS control software was replaced with the Chempiler (chapter 7.3). After essentially re-building the entire platform from scratch, the setup was finally reliable enough to carry out the ambitious task of synthesising diphenhydramine hydrochloride without human interference.

8.1.7 REBUILDING THE PLATFORM AND STARTING ANEW

The Backbone of the old platform was replaced with newly built pumps and valves of the latest design, running the new firmware. All reagent inlet ports were fitted with non-return valves to avoid reagent cross-contamination. The pumps were fitted with 25 mL syringes to make transfers of larger volumes more efficient. The placement of the modules and reagents on the Backbone was revised. Reagents that could react with each other were kept on separate valves. Transfer lengths were minimised where possible. Ether, isopropanol and water were kept on the leftmost or rightmost valve to allow efficient cleaning of the Backbone.

The reactor flask was replaced with a pear-shaped flask and a custom heating block (see also chapter 6.2.1). This way, the different volumes of all the steps could be more

RESULTS AND DISCUSSION

easily accommodated. The separator was fitted with a reliable conductivity sensor (see chapter 6.2.3).

All procedures developed before were transcribed into ChASM code, and the steps were tested individually. Everything worked reasonably well, so a full run was attempted. However, the aqueous workup of the Grignard reaction proved troublesome.

Unfortunately, the ammonium chloride didn't seem to be acidic enough to dissolve the $\text{Mg}(\text{OH})\text{Br}$ formed in the quench, leaving behind solid residues and complicating phase separation (Figure 75).

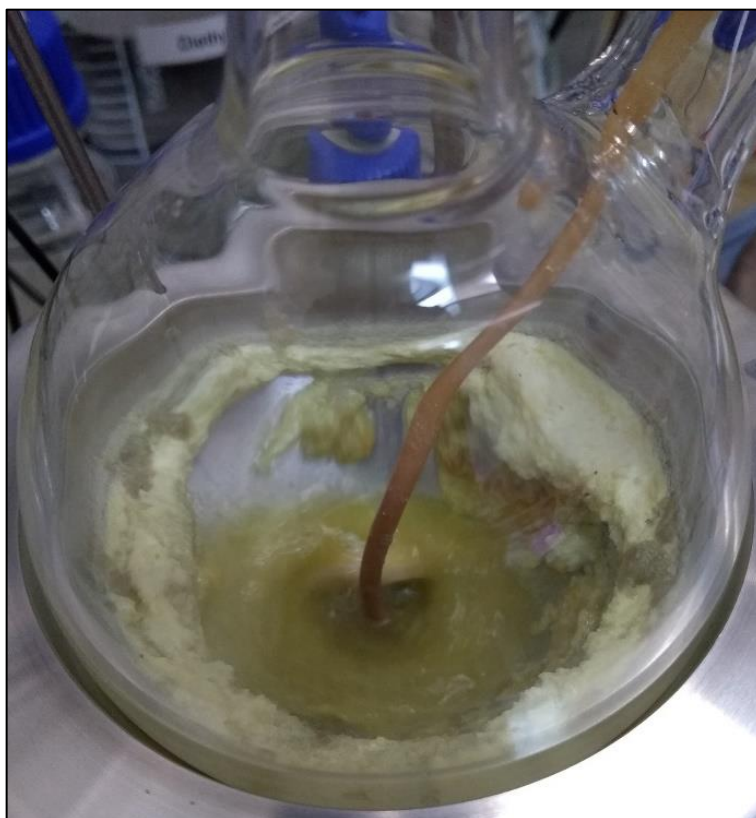


Figure 75: Solid residues left after quenching the Grignard reaction with saturated ammonium chloride.

Thus, it was ultimately decided to quench the reaction with dilute hydrochloric acid instead. The product contains no acid labile groups, and elimination of the alcohol is impossible as it is neighbored by two sp^2 carbons. Thus, there was no reason to believe that using a stronger acid for the quench would be in any way detrimental.

As expected the hydrochloric acid dissolved all precipitate and formed two clear layers.

It was also found that the magnetic stirring in the reactor did not mix the layers very efficiently. A shallow vortex was formed, and the phase boundary bobbed around a bit, but the intimate contact between the two phases that shaking them in a separatory funnel would produce could not be achieved. To address this shortcoming, a computer controllable overhead stirrer was mounted into the liquid/liquid separator, allowing vigorous mixing of the two layers. Figure 76 shows the quality of the extraction using hydrochloric acid for quenching and overhead stirring.



Figure 76: Separation of the phases during the workup of the Grignard reaction.

The previously encountered issues regarding the precipitation of diphenhydramine hydrobromide during the workup of the Williamson step resurfaced as well. After several more attempts to run a full sequence, the p-xylene eventually ran out and it

RESULTS AND DISCUSSION

was decided to substitute it for toluene. It is unclear if this change in solvent exacerbated the problem, or if in previous, successful attempts were merely a matter of luck. In any case, severe precipitation issues upon addition of water or hydrochloric acid were experienced. When the aqueous phase was added while the reaction mixture was still warm (30°C), the salt would initially dissolve, but crash out upon cooling (Figure 77).



Figure 77: Diphenhydramine hydrobromide crystallising upon addition of water. Left: catastrophic blockage of the reactor inlet. Right: blockage of the separator.

However, if the reaction mixture was cooled to room temperature before adding the aqueous layer, the gelatinous second layer that was formed in the course of the reaction would become so highly viscous that the magnetic stirring bar was trapped, and no mixing occurred (Figure 78).



Figure 78: Williamson reaction mixture after cooling and addition of water. The bottom layer is gelatinous diphenhydramine hydrobromide with the stirrer bar trapped inside. The middle layer is water, the top layer is toluene.

Ultimately, the workup protocol had to be adjusted. Instead of adding water or hydrochloric acid to the reaction mixture, equimolar aqueous sodium hydroxide was added to the reaction mixture. This would neutralise and dissolve the diphenhydramine hydrobromide, thus liberating the free base. After separation of the layers, the organic layer could be extracted with aqueous hydrochloric acid thrice as usual. The aqueous layers were then combined and neutralised with more sodium hydroxide and extracted with three portions of diethyl ether. The overhead stirrer ensured efficient extraction, and the layers were always clearly delineated. This adjusted protocol finally enabled us to reliably isolate the diphenhydramine free base.

RESULTS AND DISCUSSION

With robust workup procedures for the troublesome steps in hand, the final hydrochloride precipitation was tackled. The filter module was now attached to the inert gas system and kept under argon. Also, further flushing steps were added before commencing the precipitation to ensure no impurities were introduced into the mixture. Those precautions, together with the high purity of the free base, ensured a smooth precipitation.



Figure 79: Crude diphenhydramine hydrochloride obtained at the end of the fully automated synthesis.

Having finally solved all problems, 18 months after the very first experiment, the four-step synthesis of diphenhydramine hydrochloride had thus been performed twice under full automation without human intervention. The syntheses yielded 56% and 70%, respectively, of crude, beige powder. This corresponds to an average yield

RESULTS AND DISCUSSION

of 87% and 91% per step, respectively. Why those two yields were significantly different could not be determined. The NMR of the product was remarkably pure, with the main impurity being dimethylaminoethanol hydrochloride.

8.1.8 ADDING PURIFICATION AND CLEANING

As mentioned previously, meticulous cleaning of the Backbone was found to be very important. The 25 mL syringes used in the new platform, together with the vastly more efficient transfer routine (see chapter 7.3.5) now enabled us to clean the modules as well by filling them with washing solvents and stirring the solvent for a while, followed by emptying the solvent to waste. Usually the reactor and separator were cleaned with water, isopropanol and diethyl ether after every step. At the end of the entire sequence the rotavap flask was also cleaned, followed by another flush of the Backbone, in order to immediately ready the system for another reaction. This procedure worked surprisingly well, leaving the modules in pristine condition after every step (Figure 80).



Figure 80: Automated Liquid/Liquid Separator after automated cleaning, ready for the next step.

At the outset of automating this synthesis, further purification of the product was not intended as it was thought to be too complicated at the time. However, the capabilities of the platform had increased significantly since its humble beginnings. Crucially, the syntheses of rufinamide and sildenafil described in the following chapters forced us to develop a temperature-controlled filtration module. This jacketed filter vessel brought a recrystallisation of the crude diphenhydramine hydrochloride within reach.

Ahmadi *et al.* recrystallised their product from isopropanol.¹⁴⁰ Refluxing liquid inside the filter as would be required for a proper recrystallisation was seen critically, since there was no feasible way of adding a condenser. However, Jensen *et al.*⁹¹ reported that, in their flow setup, they only heated the crude product in isopropanol (B.P. 82.6°C) to 60°C, followed by crystallisation and filtration.

This procedure was tested with a batch of material obtained in previous, successful runs. Sure enough, the material dissolved rapidly at 60°C, but crystallisation was sluggish. Curiously, if the mixture was cooled under stirring, the pure product would readily crystallise as fine powder which could easily be collected by filtration.

Thus, with a proven procedure in hand, the recrystallisation was tested on the platform, again using previously obtained crude product. To that end, the filter module was replaced with a jacketed filter (Figure 50 right). The recirculation chiller previously servicing the rotary evaporator was fitted with a solenoid valve (see chapter 6.2.7 for more details) and connected to the filter vessel. This way, the user could switch between heating or cooling the rotary evaporator, or the filter.

The filter was then charged with isopropanol through the bottom port. Isopropanol was already present in the setup as cleaning solvent, thus no additional input was required. The mixture was then heated to 60°C, followed by slow cooling to room temperature under stirring. The product dissolved and crystallised as was expected. To ensure maximum recovery, the mixture was then cooled to -20°C and stirred for five minutes. The isopropanol was subsequently removed through the bottom port. This had to be done very slowly due to the high viscosity of isopropanol at that

RESULTS AND DISCUSSION

temperature, otherwise the pump would simply cavitate. To wash the precipitate, fresh isopropanol was added through the bottom port, the mixture was stirred at -20°C for another five minutes to allow temperature equilibration and filtered off once more.

In order to dry the product, the filter module was fitted with a vacuum valve (see chapter 6.2.7 for more details). The valve was switched to vacuum, and the filter was heated to 60°C for fifteen minutes to dry the filter cake. Eventually, the filter was cooled to room temperature again to allow convenient collection of the product (Figure 81).

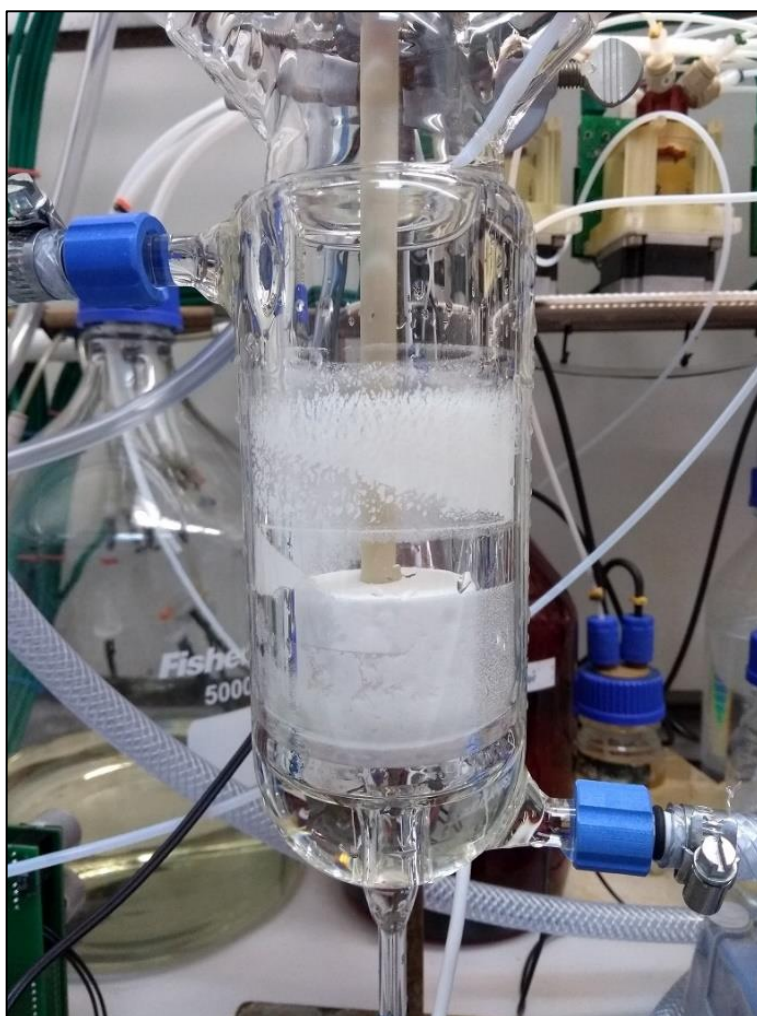


Figure 81: Jacketed filter module containing pure diphenhydramine hydrochloride after recrystallisation.

RESULTS AND DISCUSSION

This procedure was exceptionally successful. An NMR spectrum of the recrystallised product was compared with an NMR spectrum of pure ($\geq 98\%$) diphenhydramine hydrochloride obtained from Sigma-Aldrich, and except for a small amount of residual isopropanol the spectra were indistinguishable (see appendix I.I).

With a proven purification procedure in hand, the fully automated procedure was successfully repeated once more, yielding 58% of pure diphenhydramine hydrochloride (an average of 87% per step). A manual repetition of the new synthetic procedure yielded 68% overall or 91% per step in comparison. While the overall yield of the automated synthesis is ten percentage points lower than the manual execution, the yields are still seen as comparable because of the way yields compound over multiple steps. The average yields per step (87% vs. 91%) are very similar.

RESULTS AND DISCUSSION

8.2 SYNTHESIS OF RUFINAMIDE (2)

8.2.1 PREREQUISITES AND INITIAL WORK

The antiepileptic drug rufinamide (**2**) was first reported in 1986¹⁵⁰ and gained FDA approval as medication against certain types of epilepsy in 2008.¹⁵¹ As a triazole derivative, it is most commonly prepared via a click reaction between the corresponding azide and alkyne. Figure 82 shows a synthetic route patented by Kankan *et al.*¹⁵²

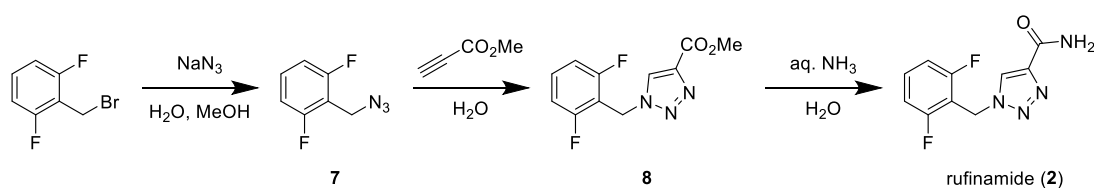


Figure 82: Synthetic sequence for rufinamide, adapted from Kankan *et al.*¹⁵²

Dr. Jaroslaw Granda manually replicated the synthesis and found it to work as published. Based on those initial findings, he built a smaller synthesis platform consisting of four Backbone units, a reactor module, and a filter module, and proceeded to automate the synthesis under my guidance.

8.2.2 AUTOMATION ON THE SMALL PLATFORM

The synthesis commences with the azide formation using 2,6-difluorobenzyl bromide and an aqueous solution of sodium azide at 75°C for 12 hours. The reactor was manually charged with the solid difluorobenzyl bromide, followed by automated addition of an aqueous solution of sodium azide and heating to 75°C. The whole step proceeded uneventfully.

In the initial attempts, the subsequent triazole click and the amide formation were performed in the reactor vessel as well, but pumping the resulting suspension of rufinamide in water into the filter module was found to be unreliable. Therefore, the difluorobenzyl azide solution was instead transferred into the filter module and the subsequent steps were performed in the filter. To that end, the filtration module was

equipped with a jacketed filter (Figure 50 right) to allow heating of the reaction mixture.

Addition of methyl propiolate solution and heating to 65°C initiated the triazole click. Subsequent addition of ammonia solution and further heating at 75°C for another 12 hours led to precipitation of the target compound. Filtration followed by three washes with water yielded pure rufinamide at $46 \pm 4\%$ isolated yield, which is slightly better than the manual synthesis (38%).

Due to the smooth operation and short overall reaction time, the synthesis was repeated multiple times to gauge the reproducibility of the yield. Table 4 shows the yields obtained on the small platform.

Table 4: Yields obtained for the automated synthesis of rufinamide.

Run	1	2	3	4	5	6
Yield [g]	4.24	4.20	3.73	3.42	4.35	4.12
Yield [%]	50	49	43	39	51	48

8.2.3 TRANSFERRING THE CODE TO ANOTHER PLATFORM

Due to the simplicity of the synthesis, it could be developed on the smaller platform, which lacked an evaporation and liquid/liquid separation module, and employed different stirrers and a different recirculation chiller. To demonstrate the power of the Chempiler software, the same ChASM file was then executed on the “full scale” platform used for the synthesis of diphenhydramine hydrochloride (see chapter 8.1).

A graph was drafted for the bigger platform, placing the reagents roughly on the same ports as on the smaller platform. The connectivity of the modules was reused from the diphenhydramine development without changes. The ChASM file that successfully produced rufinamide on the smaller platform was then run without changes.

On the first try, old reagent solutions from previous syntheses on the smaller platform were used. The synthesis worked without problems, but only yielded 29%

which was significantly lower than on previous tries. The purity by NMR was comparable with previous runs.

Thus, the reagent solutions were suspected to have degraded. All reagents were discarded, fresh solutions were prepared, and the synthesis was repeated. This time, the sequence yielded 44% which is within one standard deviation of the previously obtained yields.

This finding also suggests that the experienced variation in yields might be a result of reagent degradation. Since the reagents were not discarded and freshly prepared for every try, some runs that were conducted with older reagents would consequently yield less product than the ones that had largely fresh reagents loaded.

Overall, the simple one-pot procedure was a well-conceived synthesis pathway and had little potential for automation problems. Still, it was gratifying to see that not only could the whole synthesis be automated smoothly, but also be transferred to another physical setup with ease.

RESULTS AND DISCUSSION

8.3 SYNTHESIS OF SILDENAFIL (3)

8.3.1 PREREQUISITES AND INITIAL WORK

Sildenafil (**3**), better known under its brand name VIAGRA[®], was first reported in 1992 by Pfizer.¹⁵³ Famously, it was first investigated as cardiovascular medication, but then showed to be a potent agent for the treatment of erectile dysfunction,¹⁵⁴ and subsequently became a blockbuster drug. In 2000, Dale *et al.* published their development of the commercial synthesis route (Figure 83) as a case study.¹⁴¹ This paper gave a detailed account of the synthesis, so manual replication was commenced. The original scheme is a convergent synthesis, preparing pyrazole **11** in two steps, a nitration and subsequent reduction of the nitro group. However, pyrazole **11** was commercially available, so it was decided to avoid the hydrogenation step. That said, tackling catalytic hydrogenations and automating the pyrazole preparation is certainly a worthwhile goal in any future work.

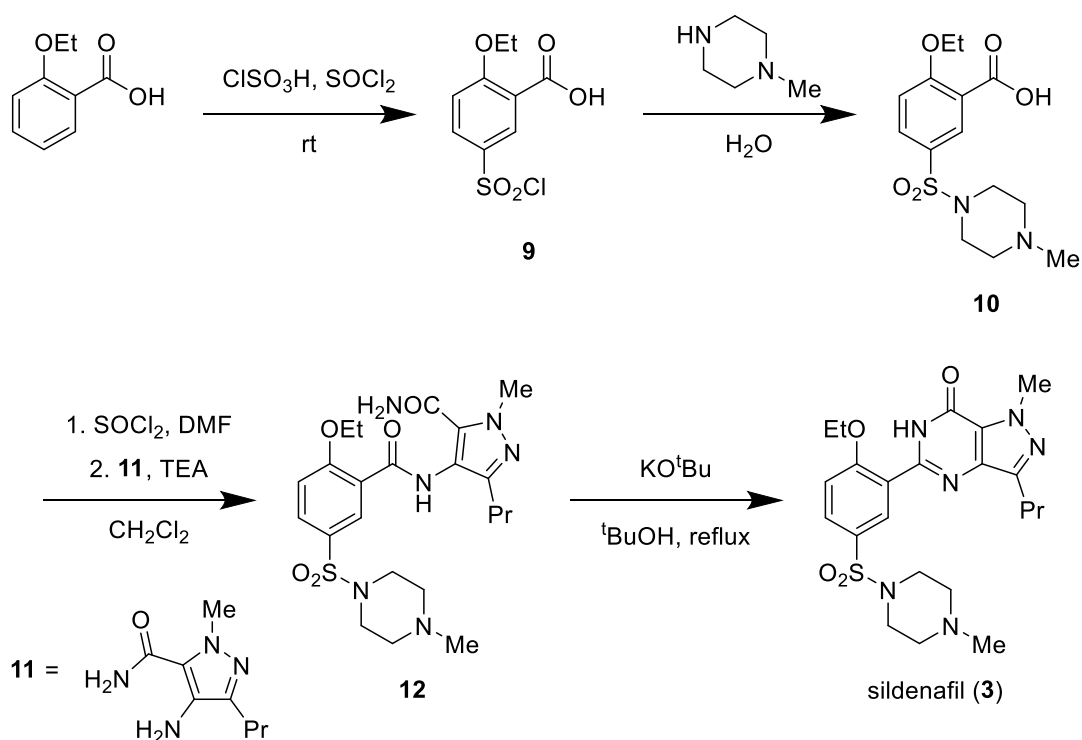


Figure 83: Synthetic sequence for sildenafil, adapted from Dale *et al.*¹⁴¹ The original procedure reported in the paper called for carbonyldiimidazole (CDI) as coupling reagent for the third step.

Sildenafil was chosen as a target molecule while the development of the diphenhydramine synthesis was still work in progress. This decision coincided with the arrival of a Master's student, Jakob Wolf, so the development and automation of the sildenafil synthesis was assigned to him. He completed his project under my direct supervision and guidance. He performed the lab work, while problems were addressed together in a cooperative fashion.

8.3.2 MANUAL REPLICATION OF THE REPORTED SYNTHESIS

The synthesis commences with a chlorosulfonation in chlorosulfonic acid and thionyl chloride. When handling it using standard disposable polypropylene syringes it quickly became apparent that the corrosive nature of chlorosulfonic acid was understated in the safety data sheet. The acid corroded the syringe, leaked past the plunger, and attacked the fume hood surface, leaving a shallow hole. In subsequent tries, the chlorosulfonic acid was handled with the same glass and PTFE syringes used in the syringe pumps, which proved to be resistant against attack.

The temperature of the water for the quenching of the reaction mixture was also found to be very important. Quenching in ice water produced near quantitative yields, while quenching in room temperature water decreased the yield below 50%. The reason seems to be that the water-insoluble sulfonyl chloride formed undergoes hydrolysis to the water-soluble sulfonic acid, which is washed away during filtration. If the quench is performed near 0°C with plenty of ice to take up the reaction heat produced by the highly exothermic hydrolysis of the remaining chlorosulfonic acid and thionyl chloride, the rate of the sulfonyl chloride hydrolysis is slow enough to recover most of the product. This finding was a key incentive for the design of the jacketed filter module (see chapter 6.2.4), which has since proven to be a very useful addition to the system.

The subsequent sulfonamide formation with *N*-methylpiperazine proceeded smoothly. However, the crystallisation of the product would only work after seeding with product. It precipitated spontaneously once, but after this one serendipitous event it never again happened without seeding. Dale *et al.* mention similar woes in

their discussion, stressing the importance of seeding. In fact, they stipulate that their initial, more complicated route *via* a double salt of the compound was abandoned only after a similarly serendipitous crystallisation event was observed. Later experiments with antisolvents, cooling below 0°C, and different stirring rates all failed to induce precipitation. Curiously, even the rough surface of a sintered glass plate, or even unreacted starting material which was sometimes present as a solid, would not initiate crystallisation. Only seeding with the product would reproducibly yield crystalline product.

Furthermore, the amount of *N*-methylpiperazine was found to be crucial, as too much or too little piperazine would also compromise the crystallisation. Dale *et al.* talk about adjusting the pH to the isoelectric point of the zwitterionic compound in their discussion, however they fail to mention where this point lies, and make no further mention in their experimental procedures. A decision was made against adjusting the pH as this would require significant instrumentation, and instead a proportion of piperazine was found which would guarantee crystallisation upon seeding.

With sulfonamide **10** in hand, coupling it with pyrazole **11** was attempted next. Alas, in our hands the CDI coupling reported by Dale *et al.* gave inconsistent and generally low yields. Furthermore, the CDI was poorly soluble in ethyl acetate, forcing us to increase the solvent volume to allow us to add the CDI as a solution. Then, the formation of the activated carboxylic acid proceeded sluggishly. Crucially, the subsequent coupling with the pyrazole did not seem to proceed at all. After several days, TLC and NMR showed virtually no conversion.

At some point an important observation was made: Dale *et al.* directly use the reaction mixture obtained from the hydrogenation of the nitro group (Figure 84). This hydrogenation produces two equivalents of water for every equivalent of amine, thus likely saturating the ethyl acetate with moisture. In this work, on the other hand, dry ethyl acetate was used to dissolve the dry piperazine, and the reaction was carried out under water-free conditions.

RESULTS AND DISCUSSION

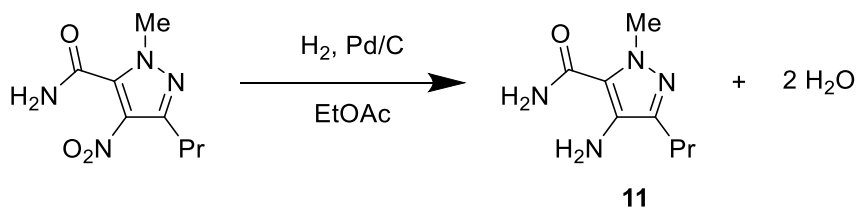


Figure 84: Hydrogenation to form the pyrazole **11**.¹⁴¹

Researchers at Astra Zeneca had previously reported similar problems related to CDI couplings when the water content was too low.¹⁵⁵ They stipulated that the coupling of the amine to the activated carboxylic acid benefits from acid catalysis, and reported the use of imidazolium hydrochloride as catalyst.

Consequently, the influence of water and imidazolium hydrochloride on the reaction was investigated. Adding small amounts of water to the pyrazole solution improved the conversion somewhat, but the yield was still low. Imidazolium hydrochloride didn't seem to have any effect on the reaction.

Ultimately, abandoning the CDI was the economical approach. Dale *et al.* used CDI on scale due to process chemical considerations, namely the ability to telescope the nitro reduction into the coupling using only one solvent, and the operational simplicity of the coupling. For this project, however, the pyrazole was bought; and the coupling was found troublesome. Thus, it was decided to activate the carboxylic acid as acid chloride instead. This approach was insofar advantageous as all reagents were liquids and therefore simple to handle in the automated setup.

Literature precedent¹⁵⁶ was found for this reaction. The published method (Figure 85) called for *N,N*-dimethylformamide (DMF) as catalyst and was performed in dichloromethane (DCM). The authors did not isolate the acid chloride, but rather used triethylamine to quench unreacted thionyl chloride as well as the hydrogen chloride byproduct. This straightforward procedure was very appealing to us.

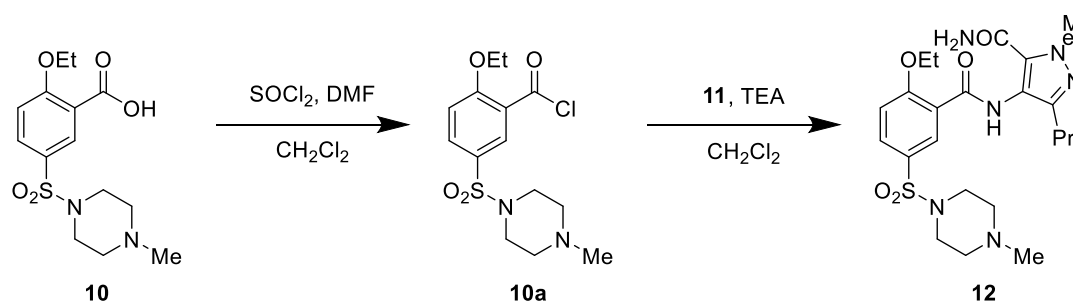


Figure 85: Acid chloride mediated amide coupling.

This coupling worked well in our hands. Two important observations were made, informing the later automation of the procedure. First, the starting material had to be dried properly as it was crystallised from water, and residual moisture would hydrolyse the thionyl chloride. Second, following aqueous workup the DCM layer contained a lot of water, which would interfere with the subsequent cyclisation. Normally, a human chemist would routinely dry any organic layers using anhydrous sodium or magnesium sulfate prior to evaporation. In the automated setup, this had yet to be implemented.

With amide **12** in hand, the base catalysed cyclisation of the purine scaffold was commenced. The original procedure called for refluxing the amide **12** with *tert*-butoxide in *tert*-butanol. Initial concerns about the high melting point of the *tert*-butanol, specifically about crystallisation inside the tubing, were unfounded as the dissolved potassium *tert*-butoxide would lower the melting point significantly, rendering a 0.5M solution liquid at room temperature.

The cyclisation reaction proceeded uneventfully. The subsequent neutralisation with aqueous HCl and crystallisation was found to work very well, yielding the title compound as pure white powder. The overall yield was 47% over four steps, which corresponds to an average yield of 83% per step.

8.3.3 SUBJECTING THE PLATFORM TO CHLOROSULFONIC ACID

With a successful experimental procedure in hand, once again the sequence was automated. My student assembled a platform initially consisting of four Backbone units, a reactor module and a filtration module. The reactor was fitted with a DrySyn SnowStorm heating and cooling block, and a 250 mL round bottomed flask. The filtration module was fitted with a jacketed filter. Initially, the SnowStorm and the jacketed filter were connected in series to allow both to be heated and cooled by one recirculation chiller.

Aspirating the chlorosulfonic acid proved more problematic than anticipated. The high viscosity demanded slow aspiration speeds, but the main problem was that it would react violently with even traces of water, producing large volumes of gas, which in turn would displace the liquid. Thus, the pump would only aspirate gas. This undesirable behaviour was counteracted partly by flushing the Backbone with acetone and diethyl ether to make sure it was very dry, and partly by aspirating multiple full syringes of chlorosulfonic acid and dispensing them directly to waste prior to charging the reactor. This way, if any traces of water are present, the chlorosulfonic acid itself would reactively dry the flow path. This strategy proved to be highly successful and allowed us to formulate the reaction mixture. The ethoxybenzoic acid (m.p. 19.3-19.5°C) could be added as a liquid as well and was flushed into the reactor with an additional portion of chlorosulfonic acid.

There were concerns about the resilience of the valve heads against the chlorosulfonic acid, but it was found that they were only attacked very slowly. Thoroughly flushing the Backbone after handling it helped prolong their life span, yet, after several runs, some of the valves had to be exchanged. The rest of the platform, however, showed no signs of deterioration due to handling chlorosulfonic acid. Replacing the valve heads with more resilient models was investigated, yet unfortunately, at the time of writing, no suitable six-way selection valves fabricated entirely from PTFE were commercially available. In the medium to long term, it might be advisable to design a custom valve head and have it manufactured to spec. For the time being, however, the current valve heads were deemed good enough.

Another complication was encountered when the reaction mixture was to be transferred into the filter module for quenching. As the water had to be cooled to near zero degrees, and the filter and the reactor were both connected to the chiller, the reaction mixture was consequently cooled to near zero as well. Unfortunately, this rendered the already highly viscous solution all but unpumpable. Thus, a solenoid valve was introduced into the chiller flow path, to enable the user to switch the chiller between either the reactor, or the filter. This way, the reaction mixture remained at room temperature, where it could be aspirated very slowly, while the filter was cooled appropriately.

During later runs, precipitate forming inside the tube from the Backbone to the reactor was encountered during the chlorosulfonation reaction, leading to blockage. This problem was not encountered in earlier runs, and the identity of the solid could not be ascertained. It was reasoned that maybe the temperature played a role, as during earlier tries the lab was heated, while during later runs in spring and summer the heating was switched off and the laboratory was noticeably cooler. However, this could not be verified experimentally.

During the reaction, chlorosulfonic acid was standing in the tube for many hours. The solid could not be chlorosulfonic acid itself which has a melting point of -80°C ,¹⁵⁷ and all decomposition products of chlorosulfonic acid that come to mind are gaseous. Thus, the identity of the precipitate remains unclear. In any event, it had to be avoided. Flushing the acid into the reactor with solvent risked losing yield after the subsequent quench due to partial dissolution of product. Using more thionyl chloride to flush in the chlorosulfonic acid was not a preferred solution as this would likely lead to the conversion of the carboxylic acid to the corresponding acid chloride and the implications for the workup and yield were unknown. Thus, flushing the tube with air was tried instead.

Flushing tubes with air had become a standard practise to empty them. The air in question was commonly drawn from one of the waste ports, after flushing the corresponding tube. Unfortunately, in this case this strategy was unsuccessful since the reactor was kept under a slight overpressure of argon, while the waste container resided at ambient pressure. Thus, whenever the pump was filled with ambient

pressure air and the valve switched to the pressurised reactor, some liquid would be pushed back into the syringe. Therefore, a small amount of chlorosulfonic acid would always remain in the tube, no matter how many portions of air were pumped in. It should be noted here that the internal pressure of the system was miniscule, but a slight overpressure had to be maintained at all times to ensure the integrity of the inert atmosphere, similar to a conventional Schlenk line. This conundrum was ultimately solved by connecting one Backbone port directly to the inert gas system via a Woulff bottle, thus providing argon at the same pressure as inside the reactor. This pressurised argon could be pumped into the reactor and would successfully displace the acid in the tube to prevent clogging.

With all those precautions in place sulfonyl chloride **9** was ultimately obtained in excellent yields.

8.3.4 INITIATING THE CRYSTALLISATION OF THE SULFONAMIDE

For the next step, the still wet sulfonyl chloride was slurried in water again, the mixture was cooled to 0.5°C, and *N*-methylpiperazine was added. A clear solution was formed after a few minutes. As expected from previous manual investigations, the product would not spontaneously crystallise at that point.

Initially, the solution was seeded with product by hand, in order to facilitate further development. Later, layering acetone over the aqueous solution was explored since acetone is an antisolvent for the product. Unfortunately, crystallisation did not occur. The stirring rate was modulated as well, hoping that either high or low rates, or no stirring at all, or abrupt changes in stirring rate would trigger crystallisation. However, the stirring rate seemed to have no effect whatsoever. Ultimately, it was decided to pump in a small volume of a slurry of product in water. To that end, a reagent bottle containing product in water was placed on a magnetic stirring plate to keep the solid suspended, and after the reaction had finished, a small volume of the suspension was pumped into the reactor. This was found to be a robust solution for the problem, guaranteeing a reliable crystallisation of the product.

8.3.5 FORMING THE ACID CHLORIDE AND PERFORMING THE AMIDE COUPLING

With sulfonamide **10** in hand, automation of the amide coupling commenced. The first challenge was to dry the precipitate to avoid immediate hydrolysis of the thionyl chloride. To that end, a vacuum system for the filter was developed (see chapter 6.2.4). The bottom outlet of the filter was connected to the central inlet of a six-way valve with a short length of tubing. One outlet of the valve was connected to the Backbone, the other outlet was connected to a Woulff bottle which in turn was connected to the laboratory vacuum supply. This contraption, together with gentle heating provided by the jacketed filter, allowed drying the solid sufficiently for the subsequent acid chloride formation.

The dry material was then slurried in dry dichloromethane which was pumped in through the bottom of the filter and the mixture was cooled to 5°C. Thionyl chloride and DMF were added, and the mixture was stirred at room temperature overnight. The reaction progress could be estimated visually as the initial slurry turned into a clear solution.

Once the acid chloride formation was complete, the reactor was charged with a solution of pyrazole **11** in DCM and triethylamine, and the mixture was cooled to 10°C. The acid chloride solution was transferred into the reactor and the mixture was stirred again overnight at room temperature to ensure full conversion.

After the reaction was complete, the mixture was quenched with water under cooling, and the layers were separated (see chapter 6.2.3 for a detailed description of the liquid/liquid separation module). At this point the organic layer was found to contain a large amount of water. Dichloromethane does not form an azeotrope with water,¹⁵⁸ so the organic phase had to be dried prior to evaporation, or else the water would interfere with the subsequent cyclisation. To that end, a drying cartridge and a switching system were developed (see chapter 6.2.7 for a detailed discussion). This cartridge allowed us to pass the wet organic phase over activated molecular sieve prior to moving it to the rotary evaporator. After evaporation to dryness, crude carboxamide **12** was obtained as off-white solid.

The NMR of the crude product contained signals tentatively assigned to sulfonamide **12a** (Figure 86). The impurity was never isolated, so the structure could not be assigned unequivocally. However, it was reasoned that unreacted sulfonyl chloride **9** remaining after the second step could react with the pyrazole to form sulfonamide **12a**. Sulfonyl chloride **9** was known to undergo hydrolysis at moderate temperatures, forming the water soluble sulfonic acid. Thus, it was decided to briefly heat the wet precipitate formed in the second step to 35°C before washing it with water. Adopting this procedure eliminated the observed side product.

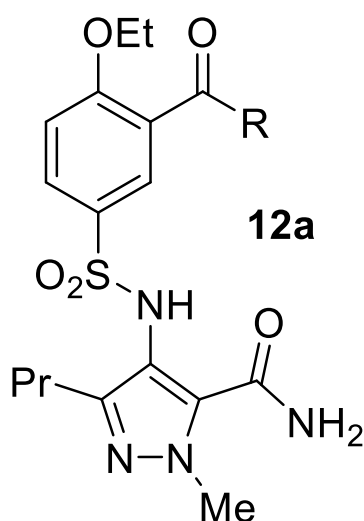


Figure 86: Proposed structure of the impurity found after the amide coupling, where R is either OH or another pyrazole moiety.

8.3.6 CLOSING THE CYCLE WITH POTASSIUM *TERT*-BUTOXIDE

The endgame commenced with transferring the crude carboxamide **12** back into the reactor. To that end, the rotavap flask containing the crude mixture was charged with a 0.5M solution of ^tBuOK in ^tBuOH. To facilitate dissolution, the heating bath of the rotary evaporator was set to 60°C and the flask was rotated in interval mode. Once the material was fully dissolved, the entire mixture was transferred to the reactor and heated to reflux for eight hours. Subsequently the reaction was cooled to 10°C and quenched with water. The mixture was transferred to the filter and neutralised with dilute hydrochloric acid, causing the title compound to crystallise.

After filtration and washing with water, the solid was dried under gentle heating and a stream of argon by switching the filter bottom to vacuum, yielding sildenafil as white powder.

8.3.7 GOING THE DISTANCE: RUNNING THE FULL SEQUENCE

As with the diphenhydramine hydrochloride, once every individual step had been successfully automated, the entire sequence was attempted under full automation.

In order to accommodate all required modules and reagents, the Backbone had to be extended to six units. Cleaning procedures were added between the steps, and the volumes of solvents and reagents were adjusted to fit the obtained yields. Breakpoints were added after every step (see chapter 7.3.8), to allow assessment of yields and purities of the intermediates.

Due to the highly corrosive nature of some of the materials handled, as well as the sheer length of the sequence, some minor problems were encountered along the way. Most were simple equipment breakdowns which could be fixed easily. Some issues were related to degradation of the valve heads, which was usually fixed by exchanging it for a fresh one. None of the issues encountered posed a significant, conceptual obstacle warranting further discussion.

After fixing all encountered errors, eventually the entire sequence was completed with breakpoints after every step under full automation, end-to-end, yielding 46% of pure sildenafil over four steps. This corresponds to an average yield of 82% per step and is consistent with the yields obtained by hand.

Emboldened by this positive result, the breakpoints were removed, and the synthesis was started once more. This time 44% (81% per step) yield were obtained, which is entirely comparable to the previously obtained 46%, as well as the 47% obtained manually. The purity by NMR was also comparable to that of the manually produced sildenafil (see appendix I.I).

RESULTS AND DISCUSSION

CONCLUSIONS AND FUTURE WORK

The development of this synthesis platform has been a challenging project. The strong interdependence between the hardware performing the chemical operations, the chemistry performed using the hardware, and the software controlling the platform mandated parallel development in all three areas, which was a taxing experience at times. The existing hardware and software solutions were suffering from severe problems and required substantial improvement.

The optimisation of the pumps and valves in particular required a disproportionately large amount of effort, especially considering that the devices, at the end of the day, are not really superior to commercially available solutions by a wide margin, if at all. However, there is some merit to the wider concept of developing a framework of standardised hardware using a common, interchangeable interface, and the exercise was of considerable educational value to me, so the effort was not entirely wasted.

Perhaps the biggest challenge was to learn how to do chemistry in an entirely new and unfamiliar way. Many operations that are second nature to a skilled organic chemist turned out to be fiendishly complicated when performed on the platform rather than by hand. Yet once the capabilities and limits of the platform were well enough understood, and standard operating procedures were developed, the automation of new chemistry became relatively frictionless.

In summary, the multistep syntheses of three active pharmaceutical ingredients in batch were fully automated at yields and purities entirely comparable to those obtained by a skilled synthetic chemist by hand. Modules for the unit operations of mixing under heating or cooling, liquid/liquid separation, evaporation and filtration were developed, as well as a number of auxiliary modules. Those unit operations are anticipated to be sufficient to perform a large fraction of organic synthesis. Furthermore, a modular software package to control the platform, and a rudimentary scripting language to describe chemical operations were developed.

The technology is already starting to spread. At the time of writing, a total of four platforms have been built within the group, and several researchers are performing

experiments on them. Furthermore, collaborators from both academia and industry have expressed interest in the platform.

The key novelties of the platform are the unit operation centric view, the modularity, and the software architecture. By looking at the physical unit operations being performed in the course of a synthesis and realising that those operations tend to fall into one of only very few classes regardless of the nature of the chemical transformations, a universal automation strategy for in principle all of organic chemistry could be devised. Unlike many other previously reported solutions which tend to focus on isolated aspects of individual problems, this platform encompasses the entirety of the synthetic process, from the reaction over workup and isolation to purification.

By strictly segregating the hardware into modules dedicated to a single class of unit operations, and by developing an efficient solution for transferring material between the modules, a platform which can be reconfigured and expanded as required was created. If not yet implemented unit operations are needed, a corresponding module can be developed and added to the platform with ease. If certain capabilities are not required in everyday use at all, the corresponding module can be omitted in order to save space and money. Furthermore, individual modules can be improved on without affecting the operation of the rest of the platform.

Evidently, this high degree of flexibility is only useful if coupled with equally modular software. The concept of separating the description of the physical layout of the platform from the synthetic operations allows the experimenter to script operations in a way that is very similar to how synthetic procedures are reported in the literature, and which is familiar to practising chemists. Furthermore, it enables the user to execute a synthesis developed on one platform to be performed on another platform, even if the employed hardware and the layout of the platform are very different. This innate portability could potentially benefit the larger community, as syntheses can be curated and distributed online, version controlled, or even developed collaboratively between multiple contributors in a fashion similar to open source coding projects.

Hopefully this work is just the beginning. The successful proof of principle herein demonstrated opens many avenues for further development in the areas of hardware, unit operations, software, and chemistry. In terms of hardware, one of the most pressing issues is the continued development (or replacement) of the pumps and valves. As mentioned earlier, the control boards urgently need attention. The valves would benefit from a new type of valve head which is more resistant to chemical attack. The reliability of the pumps could be improved by adding position sensing and stall detection.

Yet, for all the shortcomings of the pumps and valves, the idea of a single interface standard for lab equipment is worth exploring further. Power over Ethernet will likely not be the technology used as it only delivers DC voltage at one level and relatively low wattage. Trying to power a hotplate stirrer through PoE is likely futile. However, a novel interfacing standard providing data communication, a DC source capable of delivering variable voltages of 5-48 V based on negotiation, and a 230 V AC source capable of delivering around 10 A combined in one cable, fitted with ATEX compliant connectors, could replace the conventional power lead. A fume hood could feature a built-in data hub and power supply servicing a number of outlets for this new standard. The entire fume hood could be connected to a PC, or contain a computer itself, allowing the experimenter to control the equipment safely from outside via for example a touch screen. If the communication protocol is standardised as well and if equipment manufacturers start using this interface, such an intelligent fume hood would unlock a wide range of possibilities.

Regarding the unit operation level, there are two fundamental directions for improvements. One is the development of entirely new modules. Here, one of the most useful additions would definitely be a solid handler. This work focused on reactions that used liquid or dissolved reagents, and solids were charged by hand at the beginning. While this is not a tremendous limitation, the ability of adding solids automatically would be valuable. Unfortunately, constructing a contraption that works for a wide range of solids under a wide range of conditions is not a straightforward task. Preliminary tests were performed on a very simplistic solution with very limited success. Other desirable capabilities include a module for the

introduction of gases such as hydrogen, a chromatography module, or the addition of flow reactors.

Regarding the improvement of the existing modules, several ideas come to mind. First of all, an improved washing system could be beneficial. The current mode of pumping solvents into and out of the modules using the Backbone is slow and does not always reach all internal surfaces. A separate system for cleaning could utilise reduced pressure and solenoid valves similar to systems reported in the literature, and nozzles shaped to direct the solvent onto the inner walls. The reduced pressure could then also be used for drying the modules.

The reactor module in its current form could feasibly be replaced by a jacketed glass reactor with an overhead stirrer and a bottom drain valve. Such reactors are commercially available from a number of manufacturers and would offer superior temperature control and agitation. The bottom drain valve would also make emptying it easier. Feasibly, a system like MettlerToledo's OptiMax¹¹¹ could be integrated with the overall architecture and replace the current pear shaped flask. Using common laboratory glassware has its merits, but so does using appropriate technical solutions.

The rotary evaporator in its current form is working well, but the inability to perform evaporations autonomously under sensor control is a clear limitation. Appropriate sensors could be developed in house, or existing solutions could be adapted. Büchi, for example, offers a solution capable of fully autonomous distillation¹⁵⁹ which has even been evaluated in the course of this project, but an agreement for joint development could not be reached.

Furthermore, all current modules would greatly benefit from liquid level sensors. The current dead reckoning approach to volume tracking works reasonably well, but real time feedback would in any case be preferable.

The software in its current form is at a relatively completed state and running stable. All major components perform satisfactorily, so any development here would be focused on details. One aspect which could potentially be improved is the ChASM. While it has enabled us to automate those syntheses, it is still a relatively limited

CONCLUSIONS AND FUTURE WORK

language. Developing it into a complete scripting language or replacing it with a more adequate solution will become necessary at some point in the future.

Perhaps the largest potential for future work is in the chemistry. More and different syntheses should be performed on the platform, and the library of standard operating procedures should be expanded. For this work three active pharmaceutical ingredients were chosen merely because their syntheses were well described and reproducible. However, any small molecule could and should be made using the platform. Furthermore, the iterative syntheses described in chapter 1 could all be feasibly performed by the platform as well. While dedicated synthesisers will likely always outperform such a general setup in their own class, being able to perform peptide, DNA, RNA, and oligosaccharide synthesis as well as iterative cross coupling AND general small molecule synthesis in one setup would be of great value to laboratories with broader interests or constrained budgets.

To summarise, the author truly believes this platform could become a transformative enabling tool in synthetic chemistry, boosting the productivity of bench chemists, increasing lab safety and greatly improving reproducibility, finally advancing the field of synthetic chemistry into the 21st century.

CONCLUSIONS AND FUTURE WORK

EXPERIMENTAL

9 CHEMICALS AND INSTRUMENTATION

Solvents and reagents were used as received from commercial suppliers unless otherwise stated. Magnesium grit was activated prior to use by heating to 150 °C under a gentle stream of argon for 15 minutes. All NMR measurements were performed with a Bruker Avance III HD 600 spectrometer operating at 600.1 and 150.9 MHz for ^1H and ^{13}C , respectively. Unless otherwise noted, the samples for NMR experiments were prepared in CDCl_3 . Spectra were collected at 298 K, and chemical shifts are reported in ppm relative to TMS (^1H) or residual solvent (^{13}C) (multiplicities are given as s: singlet, d: doublet, t: triplet, q: quartet, m: multiplet, with coupling constants reported in Hz). The spectra were processed with the Bruker Topspin 3.5 software package.

HRMS was analysed on a Fusion Lumos Orbitrap (Thermo, San Diego, CA, USA) by direct infusion at a flow rate of 5 $\mu\text{l}/\text{min}$. MS1 Data was acquired at an Orbitrap resolution of 120000. The heated electrospray ion source was heated to 350 °C with 3.5 kV applied to the emitter. A total of 10 MS1 scans were acquired per compound. Raw data files were converted to text files using MS Convert.¹⁶⁰

In the automated syntheses, liquids were transferred at 50 mL/min unless stated otherwise. Volatile liquids like diethyl ether were aspirated at 20 mL/min to avoid cavitation, and then moved at 50 mL/min unless stated otherwise. Precise rates for every step can be found in the code used (available from the Cronin Group upon request).

10 COMPUTER CONTROLLED INSTRUMENTATION

The following paragraphs describe the hardware modules used in the various platforms in their most recent configuration. For previous versions, development history, and design rationale refer to chapter 6.

10.1 REACTOR MODULE

The reactor (Figure 87) consists of a commercially available two-necked, pear-shaped flask (DIN 12383) equipped with a Condensyn air condenser (Asynt Ltd. UK) or a Findenser waterless condenser (Radleys, UK). Heating and stirring is accomplished using an IKA RET® control-visc computer controllable stirrer hotplate (IKA® Werke GmbH & Co. KG, Germany) and a custom manufactured aluminium heating block for the pear-shaped flask. A 1/8" O.D. PTFE tube is held in place by a ground glass joint to GL18 thread adapter with a GL18 screw cap and insert (DURAN GROUP, Germany) so that its end reaches the bottom of the flask. A slight argon overpressure is maintained by the inert gas system (see chapter 10.5).

For the sildenafil synthesis, cooling of the reactor is required as well. To that end, a DrySyn SnowStorm ONE (Asynt Ltd. UK) and a 250 mL round bottomed flask are used. In conjunction with a recirculation chiller the DrySyn SnowStorm offers a temperature range of -30°C to 160°C and allows precise temperature control of the reactor.

EXPERIMENTAL



Figure 87: Photo of the reactor module.

10.2 AUTOMATED LIQUID/LIQUID EXTRACTION MODULE

The conductivity sensor consists of two lengths of 304 stainless steel tubing (1/8" O.D., 2.1mm I.D., Supelco Analytical), and an assortment of standard fluidic fittings and unions. An exploded view and a bill of materials can be found in appendix II. The sensor is connected to a custom-made separating funnel with a B45 ground glass joint at the top and two B14 side arms. Instead of a tap it has a glass ¼-28 UNF male threaded connector fitted to the bottom. An Arduino Due is used to read the sensor via a simple voltage divider circuit. A 1/8" O.D. PTFE tube is suspended by a ground glass joint to GL18 thread adapter with a GL18 screw cap and insert (DURAN GROUP, Germany). To facilitate efficient extractions through thorough mixing, a computer controlled overhead stirrer (Heidolph RZR2052 control by Heidolph Instruments GmbH & CO. KG, Germany; or IKA microstar 7.5 control by IKA® Werke GmbH & Co. KG, Germany) with a PTFE stirrer shaft and blade is fitted above the separator. A three-necked round bottomed flask with a tube fitted with a GL18 screw cap and insert (DURAN GROUP, Germany) is suspended over an IKA RET® control-visc computer controllable stirrer hotplate (IKA® Werke GmbH & Co. KG, Germany) as holding vessel.

EXPERIMENTAL



Figure 88: Photo of the liquid/liquid extraction module ALLEX.

EXPERIMENTAL

10.3 SOLVENT EVAPORATION MODULE

For the solvent evaporation, a computer controllable IKA RV 10 digital rotary evaporator with HB 10 heating bath (both IKA® Werke GmbH & Co. KG, Germany) and a computer controllable MD1C vario PLUS vacuum pump with CVC3000 vacuum controller (Vacuubrand GmbH & Co. KG, Germany) is modified by routing a piece of PTFE tubing through the vapour duct into the evaporation flask to pump product into and out of the flask. The receiver flask is fitted with a glass ¼-28 UNF male threaded connector and a PTFE tube (via a straight union piece and a Viton™ O ring), allowing it to be emptied *in situ*. A magnetic stirrer bar (COWIE Technology, UK) is affixed to the tube using PTFE shrink wrap. A neodymium magnet (first4magnets, UK) is mounted to a support rod, which in turn is mounted directly into the base of the rotary evaporator. To that end, an M10 hole is drilled and tapped into the die-cast aluminium base plate. The magnet is positioned above the magnetic stirring bar, so it lifted the inlet tube when the flask is lifted up (Figure 42).



Figure 89: Photo of the solvent evaporation module.

10.4 FILTRATION MODULE

The filtration unit consists of a custom-made sintered glass Büchner funnel (made in-house, see Figure 50 and appendix II) fitted with a B29 ground glass joint at the top, a B14 side arm, and a glass ¼-28 UNF male threaded connector at the bottom. The top inlet tube is suspended by a ground glass joint to GL18 thread adapter with a GL18 screw cap and insert (DURAN GROUP, Germany) while the bottom outlet tube is connected to the threaded connector with a straight union piece and a Viton™ O-ring. The top B29 joint is fitted with a cone to tubing adapter connected to the inert gas system (see chapter 10.5). The filter is suspended over an IKA RET® control-visc computer controllable stirrer hotplate (IKA® Werke GmbH & Co. KG, Germany) to allow magnetically stirring the contents. To ensure efficient magnetic coupling, the bottom outlet tube is bent at 90°, thus minimising the distance between the glass frit and hotplate stirrer. A jacketed version of the filter module is used when heating or cooling the content is required. In that case, the magnetic stirrer is replaced with a computer controllable overhead stirrer (Heidolph RZR2052 control by Heidolph Instruments GmbH & CO. KG, Germany; or IKA microstar 7.5 control by IKA® Werke GmbH & Co. KG, Germany). The jacketed version is fitted with two B14 sidearms, one for the inlet tube as described above, and the other for the inert gas.

To allow efficient drying of the precipitate, the bottom outlet of the filter is connected to the central inlet of a six-way valve. One outlet of that valve is in turn connected to the Backbone, while another outlet is connected to the laboratory vacuum system via a Woulff bottle (assembled from a 250 mL GL 45 pressure plus+ bottle, a GL connection system bottle top, a GL14 screw cap and insert, and a GL14 barbed hose connector, all from DURAN GROUP, Germany). This allows the user to switch the filter bottom between the Backbone (for liquid addition or withdrawal) and vacuum (for drying).

EXPERIMENTAL

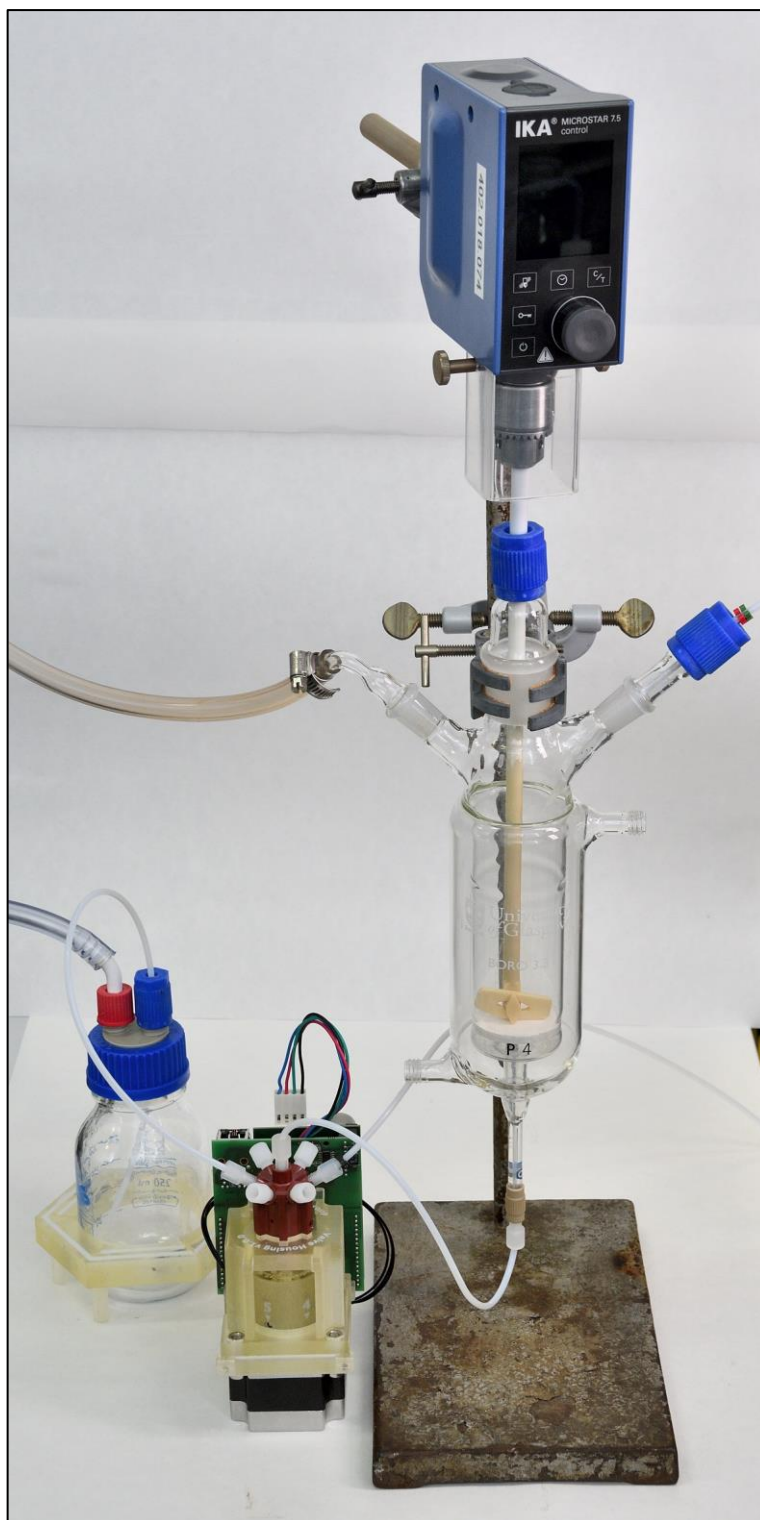


Figure 90: Photo of the filtration module.

10.5 INERT GAS SYSTEM

To supply inert gas to the various parts of the system in a modular fashion, a distribution system was assembled from pneumatic fittings and hoses. A pneumatic flow regulator (SMC Corporation, Japan) controls the flow of argon from the laboratory supply into a pneumatic manifold with two inlets and ten outlets (SMC Corporation, Japan). The outlet ports are fitted with non-return valves to avoid any cross-contamination. Those valves can then either be fitted with a barbed hose connector to connect reagent bottles to the inert gas supply or plugged shut if not in use. The second inlet is used as an outlet as well and fitted with a non-return valve. This valve is fitted with a barbed hose connector and subsequently connected to the inlet of an overpressure bubbler (Chemglass Life Sciences, USA). The outlet of the bubbler is connected to the top of the reactor and/or filter module, while the exhaust port is routed into the fume hood extract. A schematic and bill of materials can be found in appendix II.



Figure 91: Inert gas system. Left: complete view. Middle: detail view of the bubbler. Right: detail view of the manifold for the reagent bottles.

10.6 REAGENT STORAGE SYSTEM

All solvents and reagents are stored in 100 mL or 250 mL GL45 bottles (Fisherbrand, Fisher Scientific UK Ltd.) fitted with GL connection system bottle tops (DURAN GROUP, Germany), providing two GL14 threaded ports. A 1/8" O.D. PTFE tube is fitted into one port with a GL14 screw cap and insert (DURAN GROUP, Germany). For air sensitive materials the second port is fitted with another GL14 screw cap and insert (DURAN GROUP, Germany) and a polypropylene diaphragm check valve (Cole-Parmer Instrument Company, LLC; UK) which is connected to the inert gas system (see chapter 10.5) via silicone rubber or Viton™ tubes, maintaining a slight argon overpressure inside the bottle. To avoid bottles toppling over, hexagonal brackets with feet were designed and 3D printed. The brackets are mounted onto the bottles and held in place by friction. Small magnets are fitted into the sides allowing individual hexagons to be assembled into an orderly, modular rack holding the bottles in place.

EXPERIMENTAL



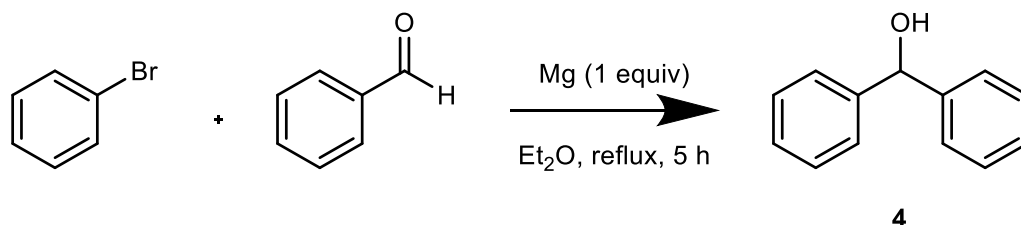
Figure 92: Reagent storage bottle. The left (black) tube is connected to the inert gas system, the right (white) tube to the Backbone.

EXPERIMENTAL

11 SYNTHESIS AND CHARACTERISATION OF COMPOUNDS

11.1 DIPHENHYDRAMINE HYDROCHLORIDE

11.1.1 DIPHENYLMETHANOL (4)



Manual:

A 250 mL round bottomed flask was charged with magnesium grit (2.5047 g, 103.1 mmol) and diethyl ether (40 mL). A dropping funnel was charged with bromobenzene (11 mL, 103.3 mmol), and a small portion was added to the magnesium grit under stirring. Once the Grignard formation was initiated, the rest of the bromobenzene was added over the course of half an hour, maintaining gentle reflux. The dropping funnel was flushed with a small amount of diethyl ether and the reaction mixture was heated to reflux for half an hour.

After cooling to room temperature, a solution of benzaldehyde (10.542 g, 99.3 mmol) in 50 mL of diethyl ether was slowly added to the reaction over the course of 20 min. After the addition was complete, the dropping funnel was flushed with a small amount of diethyl ether and the mixture was heated to reflux for four hours.

The reaction was subsequently put on an ice bath and quenched with dilute hydrochloric acid (2 M, 100 mL). The layers were separated, the organic layer was washed with 100 mL of water and evaporated to dryness, yielding 17.85 g (98%) of crude diphenylmethanol as yellow solid.

Crude ¹H and ¹³C NMR spectra were in agreement with those reported in the literature.

EXPERIMENTAL

Automated:

The reactor flask was charged manually with magnesium grit (2.50 g, 102.7 mmol) and automatically with diethyl ether (40 mL). Bromobenzene (neat, 2.95 g, 2.0 mL, 18.8 mmol) was added under stirring, followed by diethyl ether (10 mL) to ensure quantitative transfer. The mixture was heated to reflux (50°C) and stirred for 20 minutes to initiate the Grignard formation.

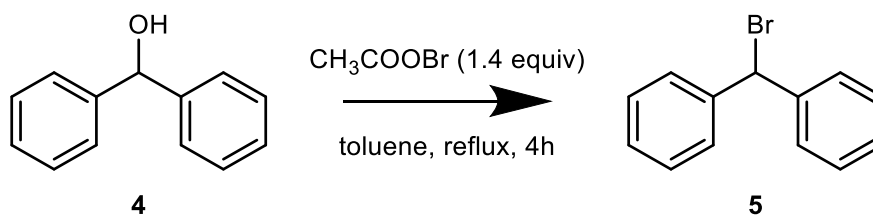
After cooling below 25°C the rest of the bromobenzene (12.75 g, 8.65 mL, 81.2 mmol) was added slowly (1 mL/min) under stirring, followed by another portion of diethyl ether (10 mL) to ensure quantitative transfer. The mixture was then stirred for 20 minutes at room temperature followed by 20 minutes at reflux (50°C).

After the Grignard formation was finished, benzaldehyde (2 M in diethyl ether, 50 mL, 0.1 mol) was added slowly (1 mL/min). The mixture was then held at reflux and stirred for 5 hours.

After the reaction cooled below 30°C, saturated ammonium chloride solution (30 mL) was added under vigorous stirring. After successful quench the mixture was diluted with 20 mL of 2 M hydrochloric acid and stirred for 15 minutes. The layers were separated and the organic layer was washed with water, transferred to the rotary evaporator and concentrated *in vacuo* (700 mbar, 30 minutes), yielding crude diphenylmethanol as a yellow solid.

Crude ^1H and ^{13}C NMR spectra were in agreement with those reported in the literature.

11.1.2 BROMODIPHENYLMETHANE (5)

Manual:

A 250 mL round bottomed flask was charged with crude diphenylmethanol (17.85 g, 97 mmol), acetyl bromide (16 mL, 216 mmol), and toluene (125 mL) and the mixture was heated to reflux for 4 hours. After cooling to room temperature, the mixture was concentrated *in vacuo*, yielding crude bromodiphenylmethane (24.16 g, 101%) as brown oil.

Crude ¹H and ¹³C NMR spectra were in agreement with those reported in the literature.

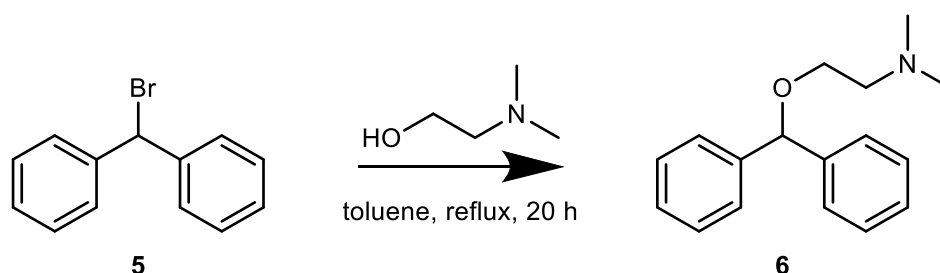
Automated:

The reactor flask was charged automatically with acetyl bromide (14.83 g, 8.94 mL, 120.6 mmol) and the lines were cleaned with toluene (3x) and air (1x). The crude diphenylmethanol remaining in the rotary evaporator flask (15.88 g, 86.2 mmol) was transferred to the reactor flask with three portions of toluene (20; 10 and 5 mL) and the mixture was heated to reflux (130°C) for 4 hours.

Upon completion of the reaction the mixture was transferred to the rotary evaporator with two portions of toluene (10 mL) and concentrated *in vacuo*, yielding crude bromodiphenylmethane as brown oil.

Crude ¹H and ¹³C NMR spectra were in agreement with those reported in the literature.

EXPERIMENTAL

11.1.3 2-(DIPHENYLMETHOXY)-*N,N*-DIMETHYLETHANAMINE (DIPHENHYDRAMINE) (6)Manual:

A 250 mL round bottomed flask was charged with crude bromodiphenylmethane (24.16 g, 98 mmol), 2-(dimethylamino)ethanol (12.5 mL, 124 mmol), and toluene (50 mL) and the mixture was heated to reflux for 18 hours.

After cooling to room temperature, the reaction was quenched with aqueous sodium hydroxide solution (4M, 25 mL) and water (25 mL). The layers were separated, and the organic layer was extracted with three portions of dilute hydrochloric acid (2M, 3x 20 mL) and water (3x 20 mL). Aqueous sodium hydroxide solution (4M, 50 mL) was added to the combined aqueous layers, and the resulting mixture was extracted three times with diethyl ether (3x 20 mL).

The combined etheric layers were concentrated *in vacuo*, yielding the title compound (22.5 g, 90%) as brown oil.

Crude ¹H and ¹³C NMR spectra were in agreement with those reported in the literature.

Automated:

The reactor flask was charged automatically with 2-(dimethylamino)ethanol (12.43 mL, 11.02 g, 123.6 mmol, 20 mL/min) and toluene (10 mL). The crude bromodiphenylmethane remaining in the rotary evaporator flask (20.36 g, 82.4 mmol) was transferred to the reactor flask with three portions of toluene (15; 10 and 10 mL) and the mixture was heated to reflux (130°C) for 20 hours.

Upon completion of the reaction the reaction mixture was allowed to cool below 40°C and diluted with toluene (25 mL). Aqueous sodium hydroxide (4 M, 25 mL) was added and the mixture was stirred for 15 minutes.

The layers were separated and the organic layer was extracted with three portions of hydrochloric acid (2 M, 3x 20 mL). The aqueous layers were combined, the organic layer was discarded.

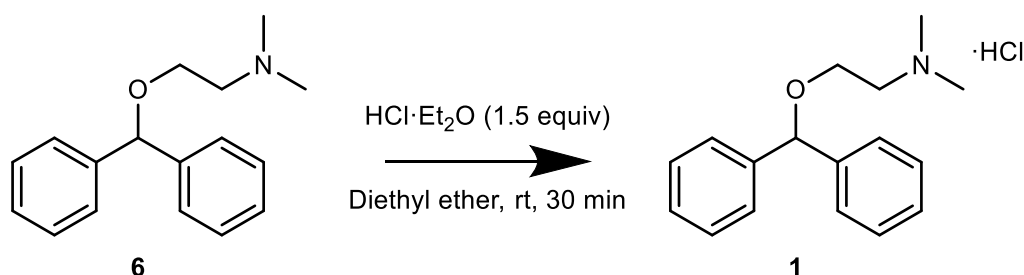
Subsequently sodium hydroxide solution (4 M, 50 mL) was added to the combined aqueous layers under vigorous stirring. The alkaline aqueous layer was then extracted with three portions of diethyl ether (3x 20 mL).

The combined etheric layers were transferred to the rotary evaporator and concentrated in vacuo, yielding crude diphenhydramine as brown oil.

Crude ^1H and ^{13}C NMR spectra were in agreement with those reported in the literature.

EXPERIMENTAL

11.1.4 2-(DIPHENYLMETHOXY)-*N,N*-DIMETHYLETHANAMINE HYDROCHLORIDE (1)



Manual:

A 250 mL round bottomed flask was charged with hydrogen chloride in diethyl ether (2M, 45 mL). A solution of 2-(diphenylmethoxy)-*N,N*-dimethylethanamine (22.54 g, 88 mmol) in diethyl ether (60 mL) was added slowly. The resulting precipitate was stirred for one hour and collected by filtration.

The crude product was slurried in 2-propanol (120 mL), heated to 70°C, and stirred until fully dissolved. The mixture was then slowly cooled to room temperature and the product was allowed to crystallise. Subsequently the mixture was kept at -20°C overnight to ensure complete crystallisation.

The solid was collected by filtration, washed with one portion of cold 2-propanol, and dried *in vacuo*, yielding the title compound as off-white powder (19.80 g, 77% for the hydrochloride precipitation only; or 68% over all four steps, corresponding to an average yield of 91% per step).

Automated:

The dead volume of the filter module was filled with diethyl ether from the bottom. Then the top of the filter was charged with etheric hydrochloric acid (2 M, 41.20 mL, 82.4 mmol). The crude diphenhydramine remaining in the rotary evaporator flask (14.03 g, 82.4 mmol) was transferred to the filter with three portions of diethyl ether (3x 20 mL) under stirring, slowly dripping it into the acid. The hydrochloride was allowed to crystallise for 30 minutes, and the solvent was subsequently removed via the bottom port. The product was then dried *in vacuo*, yielding crude diphenhydramine hydrochloride as pale-yellow solid.

Crude ^1H and ^{13}C NMR spectra were in agreement with those reported in the literature.

The crude product was automatically recrystallized from isopropanol (110 mL). The isopropanol was added through the bottom port and the suspension was heated to 60°C, yielding a clear, yellow solution. The solution was then slowly cooled to 30°C under stirring, at which point the pure product crystallized. The filter was then cooled to -20°C, and the suspension was stirred for another five minutes. Then the stirring was stopped, and the supernatant solution was withdrawn through the bottom port. The solid was washed with isopropanol (50 mL) at -20°C and dried under argon flow at 60°C for 15 minutes, yielding the title compound (16.85 g, 58% over four steps, corresponding to an average yield of 87% per step) as white, crystalline powder.

^1H NMR (600 MHz, DMSO-*d*₆) δ 10.87 (s(br), 1H), 7.43 (d, 4H, $J = 7.32$ Hz), 7.34 (t, 4H, $J = 7.65$ Hz), 7.26 (t, 2H, $J = 7.35$ Hz), 5.56 (s, 1H), 3.72 (t, 2H, $J = 5.04$ Hz), 3.33 (t, 2H, $J = 4.95$ Hz), 2.76 (s, 6H).

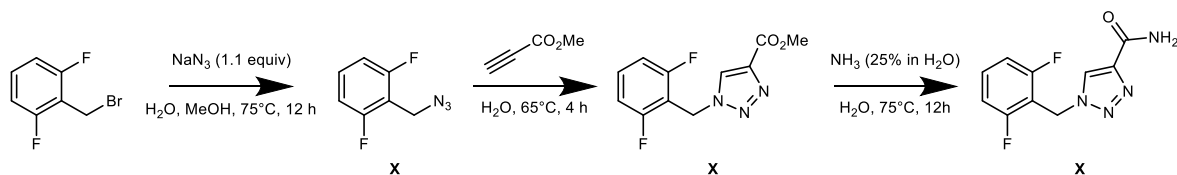
^{13}C NMR (151 MHz, DMSO-*d*₆) δ 141.86, 128.37, 127.43, 126.59, 82.85, 62.89, 55.70, 42.55.

HRESI-MS: [$\text{C}_{17}\text{H}_{22}\text{NO}$]⁺ calculated 256.1696 found 256.1685.

EXPERIMENTAL

11.2 RUFINAMIDE

11.2.1 1-[(2,6-DIFLUOROPHENYL)METHYL]-1H-1,2,3-TRIAZOLE-4-CARBOXAMIDE (RUFINAMIDE) (2)



Manual:

A 100 mL round bottom flask was charged with 2,6-difluorobenzyl bromide (2.5 g, 12.0 mmol) and sodium azide (0.845 g, 13.0 mmol, solution in 25 mL of water). The mixture was heated to $70\text{--}75^\circ\text{C}$ overnight. The formation of the product can be monitored by TLC using *n*-hexane as an eluent. After 12 h TLC showed complete conversion of the starting material. The reaction mixture was cooled to room temperature, methyl propiolate (1.07 mL, 12 mmol) was added, and the mixture was heated to $60\text{--}65^\circ\text{C}$. After 4h TLC (*n*-hexane / AcOEt 1:1) indicated complete conversion of the azide to triazole. A yellowish precipitate of the product can be also observed. After cooling to room temperature, ammonia (25% in water, 20 mL) was added and the mixture was heated to 75°C for 12 h. After cooling to room temperature, the product was filtered off, washed with water, and dried *in vacuo* at 75°C , yielding the title compound (1.11 g, 38%) as a white precipitate.

Automated:

The reactor flask was charged with 2,6-difluorobenzylbromide (7.5 g, 39 mmol), the tubing to the sodium azide bottle was primed with sodium azide solution (10.12 g in 240 mL of water), and sodium azide solution (60 mL, 39 mmol) was added to the reactor. After the addition was complete, the Backbone was cleaned with two portions of water. The reaction mixture was stirred and heated to 75°C for 12 h, and then allowed to cool below 30°C. The bottom inlet of the filtration module was primed with 15 mL of water, and the reaction mixture was transferred to the filtration module. After priming the inlet tubing, methyl propiolate (3.21 mL, 36 mmol) was added. The mixture was stirred and heated to 65°C for 4 h. After cooling to 30°C, another portion of water (15 mL) was pumped into the bottom inlet of the filtration module to purge the frit and prevent precipitation. Aqueous ammonia (25%, 60 mL) was added to the stirred reaction mixture at 20 mL/min. The reaction mixture was then heated to 75°C for 12 hours, followed by cooling to 30°C. The precipitate formed was subsequently filtered off by removing the supernatant solution through the bottom port at 5 mL/min.

The precipitate was washed with water (3x 20 mL) and dried *in vacuo* at 75°C for 4 hours. After cooling to room temperature, pure Rufinamide (46 ± 4 %, see Table 4) was obtained as a white powder.

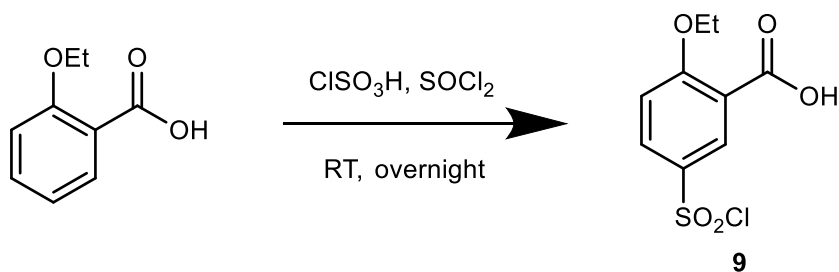
¹H NMR (600 MHz, DMSO-*d*₆) δ 8.55 (s, 1H), 7.84 (s, 1H), 7.55 – 7.49 (m, 1H), 7.47 (s, 1H), 7.19 (t, *J* = 8.0 Hz, 2H), 5.72 (s, 2H).

¹³C NMR (151 MHz, DMSO-*d*₆) δ 161.28, 160.80 (dd, *J* = 249.5, 7.4 Hz), 142.83, 131.83 (t, *J* = 10.5 Hz), 126.78, 111.97 (dd, *J* = 20.5, 4.4 Hz), 111.04 (t, *J* = 19.1 Hz), 41.20 (t, *J* = 3.8 Hz).

HRESI-MS: [C₁₀H₈F₂N₄O+Na]⁺ calculated 261.0558 found 261.0557.

11.3 SILDENAFIL

11.3.1 5-CHLOROSULFONYL-2-ETHOXYBENZOIC ACID (9)

Manual:

2-Ethoxybenzoic acid (7.53 mL, 50 mmol) was added to a mixture of thionyl chloride (3.66 mL, 50.18 mmol) and chlorosulfonic acid (13.74 mL, 206.7 mmol) while cooled on an ice-bath. The resulting solution was stirred over night at room temperature before it was quenched in 110 mL of ice-cooled water. The formed precipitate was stirred for 1h, filtered off, and washed with little cold water, yielding the title compound as white powder. A small aliquot was dried *in vacuo* to estimate the yield (12.01 g, 45.4 mmol, 90%).

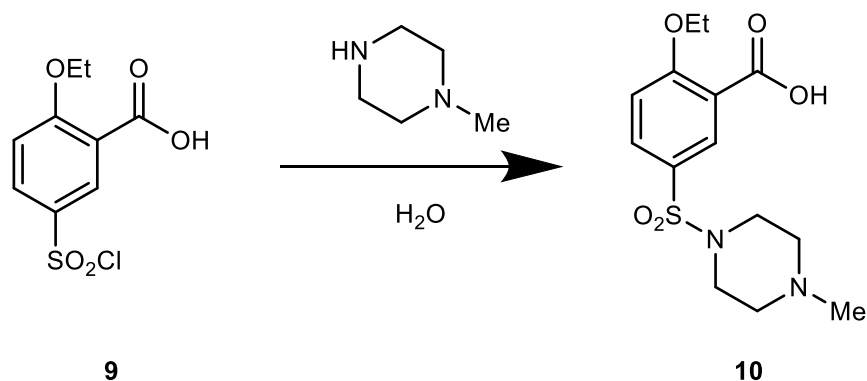
Automated:

To dry the Backbone, it was flushed with 10 mL of chlorosulfonic acid. The reactor flask was cooled to 15°C and charged with chlorosulfonic acid (10.31 mL, 154.96 mmol) and thionyl chloride (2.75 mL, 37.70 mmol). To this mixture 2-ethoxybenzoic acid (5.65 mL, 37.57 mmol) was added slowly (0.5 mL/min) which was flushed into the reactor with further chlorosulfonic acid (4 mL). The reaction was stirred for 30 min at 15°C, after which it was stirred for 12 hours at room temperature. The dead volume beneath the sinter plate was filled with 25 mL of water. Subsequently, the jacketed filter flask was filled with 31.5 mL of water from the bottom and allowed to equilibrate to 0.5 °C for 20 min. After flushing the Backbone with further chlorosulfonic acid, the reaction mixture from the reactor flask was transferred slowly (2 mL/min) to the filter flask to be quenched under vigorous stirring. After completion of the quench, the precipitate was crystallised for 1 hour, isolated by filtration and washed with two portions of water (8 mL). The title compound was obtained as a wet, white powder which was directly used in the next step without purification.

Crude ^1H and ^{13}C NMR spectra were in agreement with those reported in the literature.

EXPERIMENTAL

11.3.2 2-ETHOXY-5-(4-METHYL-1-PIPERAZINESULFONYL)BENZOIC ACID (10)

Manual:

Wet 5-chlorosulfonyl-2-ethoxybenzoic acid (11.8 g dry calculated, 44.6 mmol) obtained from the previous step was suspended in 40 mL of water and placed in an ice bath. *N*-methylpiperazine (11.4 mL, 102.8 mmol) was added and the reaction was stirred for 5 min. After seeding with a little bit of product, the solution turned cloudy. The precipitate was stirred for 2h at 0 °C, collected by filtration, washed with cold water and dried *in vacuo*, yielding the title compound as white powder (10.08 g, 30.7 mmol, 69%).

Automated:

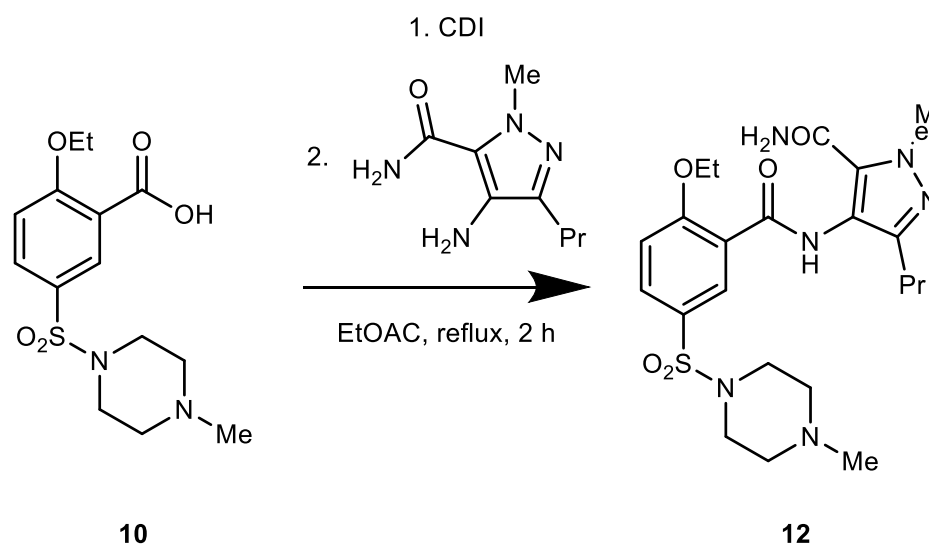
The dead volume beneath the sinter plate was filled with 25 mL of water and the wet precipitate of the previous step was suspended in water (31 mL, 0.5°C). *N*-Methylpiperazine (8.634 mL, 77.8 mmol) was added slowly (0.5 mL/min). Residues were flushed in with water (1 mL). The ensuing clear solution was seeded with a suspension of product in water (4 mL, 3 w% in HPLC grade water) after 5 min. The solution turned turbid after one minute and the solid was filtered off after 2 hours of stirring at 0.5°C. The precipitate was heated for 10 min to 35°C to hydrolyse residual starting material. After cooling back to 0.5°C the product was washed twice with water (6 mL) and dried under a stream of argon (6 hours, 50°C) to yield a white powder (8.996 g, 27.4 mmol, 73% over step one and two).

¹H NMR (600 MHz, DMSO-*d*₆): δ 7.89 (d, *J* = 2.3 Hz, 1H), 7.81 (dd, *J* = 8.8 Hz, 2.3 Hz, 1H), 7.34(d, *J* = 8.9 Hz, 1H), 4.21 (q, *J* = 6.9 Hz, 2H), 2.87 (s(br), 3.7H), 2.38 (s(br), 3.7H), 2.15 (s, 3H), 1.36 (t, *J* = 6.9 Hz, 3H).

¹³C NMR (151 MHz, DMSO-*d*₆): δ 166.1, 160.5, 132.3, 130.0, 125.7, 122.2, 113.9, 64.7, 53.4, 45.6, 45.2, 14.3.

HRESI-MS: [C₁₄H₂₀N₂O₅S+H]⁺ calculated 329.1166 found 329.1155

11.3.3 4-[2-ETHOXY-5-(4-METHYL-1-PIPERAZINYSULFONYL)BENZAMIDO]-1-METHYL-3-PROPYL-1H-PYRAZOLE-5-CARBOXAMIDE (12)



Manual:

Dry DCM (50 mL) was added to 2-Ethoxy-5-(4-methyl-1-piperazinesulfonyl)-benzoic acid (10.01 g, 30.05 mmol). The suspension was put on an ice bath and thionyl chloride (2.67 mL, 36.8 mmol) was slowly added. Subsequently, catalytic amounts of *N,N*-dimethylformamide (0.09 mL, 1 mmol) were added and the mixture was stirred overnight at 25°C.

A solution of 4-amino-1-methyl-3-*n*-propyl-1*H*-pyrazole-5-carboxamide (5.278 g, 28.96 mmol) in dry DCM (54 mL) was put on an ice-bath and triethylamine (17.1 mL, 122.7 mmol) was added. The acid chloride solution was added dropwise and reacted for 30 min. After warming the mixture to 25°C and stirring overnight it was quenched and washed with water (90 mL). The organic layer was dried over magnesium sulfate and the solvent removed *in vacuo* to give a slightly brownish solid (13.5155 g, 27.44 mmol, 90%).

Automated:

The involved parts of the Backbone and the dead volume of the filter flask were flushed with little thionyl chloride and DCM for drying. The dead volume beneath the sinter plate was filled with 25 mL of dry DCM. The sulfonamide in the filter module was suspended in dry DCM (22 mL) pumped into the filter from the bottom. At 5°C thionyl chloride was added (2.68 mL, 4.37 g, 36.7 mmol) very slowly (0.5 mL/min) and flushed in with further DCM (2 mL) to prevent reaction with DMF. DMF was added in catalytic amounts (0.09 mL) followed by DCM (2 mL) to ensure quantitative transfer. The mixture was then stirred for 5 hours at 25°C. After drying the Backbone with little thionyl chloride and DCM again, the reactor was charged with triethylamine (5 mL, 35.9 mmol) followed by a 0.5 M solution of 4-amino-1-methyl-3-*n*-propyl-1H-pyrazole-5-carboxamide in DCM (57 mL, 28.5 mmol), cooled to 10°C and basified with triethylamine (17.2 mL, 123.4 mmol). The crude acid chloride solution was added slowly (0.5 mL/min) and under stirring to the basified pyrazole solution. Upon completion of the transfer, the mixture was warmed to 25°C and stirred for 16 hours. Water (84 mL) was added for quenching after cooling back to 10°C and the mixture was stirred for 10 min. DCM (50 mL) was added to ensure complete dissolution of products. The phases were separated, and the organic phase dried over activated molecular sieve, which was washed twice with DCM (10 mL) to ensure quantitative transfer. The combined organic phases were evaporated to dryness (700 mbar, 45 min followed by full vacuum, 45 min) to yield a yellow-greyish solid. (12.2365 g, 24.78 mmol, 91%).

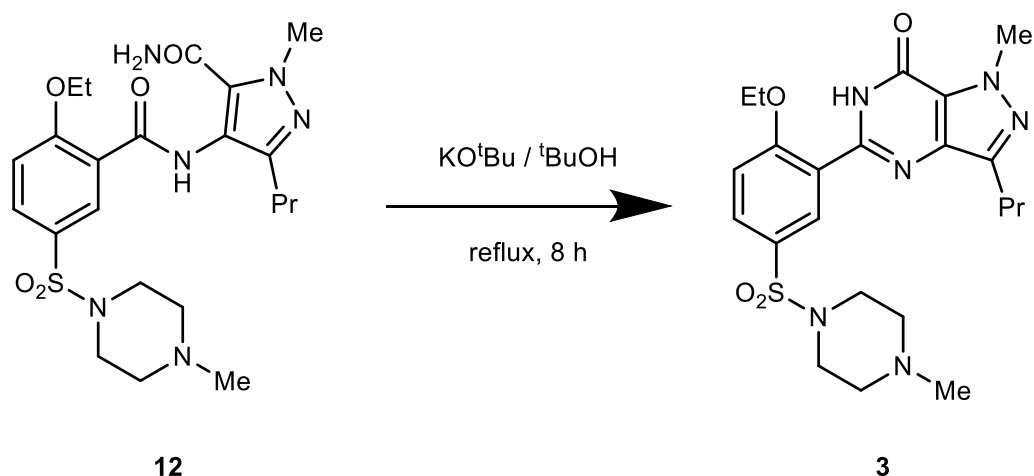
¹H NMR (600 MHz, CDCl₃): δ 9.26 (s(br), 1H), 8.63 (d, *J* = 2.3 Hz, 1H), 7.91 (dd, *J* = 8.7 Hz, 2.3 Hz, 1H), 7.67 (s(br), 1H), 7.17 (d, *J* = 8.8 Hz, 1H), 5.64 (s(br), 1H), 5.30 (s, 0,2H), 4.39 (q, *J* = 7.0 Hz, 2H), 4.06 (s, 2.8H), 3.06 (s(br), 4H), 2.54 (t, *J* = 7.6 Hz, 2H), 2.49 (s(br), 4H), 2.27 (s, 3H), 1.66 (m, 2H), 1.59 (t, *J* = 7.0 Hz, 3H), 1.3 (t, 0.4H) 0.95 (t, *J* = 7.4 Hz, 3H).

¹³C NMR (151 MHz, CDCl₃): δ 165.0, 161.5, 160.2, 146.9, 133.8, 133.2, 133.1, 128.8, 120.9, 115.2, 113.0, 66.3, 54.1, 46.2, 45.8, 39.4, 27.8, 22.5, 14.8, 14.1

HRESI-MS: [C₂₂H₃₂N₆O₅S+H]⁺ calculated 493.2228 found 493.2215.

EXPERIMENTAL

11.3.4 1-[4-ETHOXY-3-(6,7-DIHYDRO-1-METHYL-7-OXO-3-PROPYL-1H-PYRAZOLO[4,3-D]PYRIMIDIN-5-YL)PHENYLSULFOYL]-4-METHYLPYPERAZINE (SILDENAFIL) (3)



Manual:

4-[2-Ethoxy-5-(4-methyl-1-piperazinylsulfonyl)-benzamido]-1-methyl-3-propyl-1H-pyrazole-5-carboxamide (13.32 g, 27.04 mmol) was added to a 0.5 M solution of ^tBuOK in ^tBuOH (64.95 mL, 1.2 eq) and the mixture was heated to reflux for 8 hours. After cooling to room temperature 67.6 mL of water were added, followed by dropwise addition of 67.5 mL of dilute HCl (0.447 M, 30.2 mmol) over 2 hours. The precipitated product was stirred and allowed to crystallise at pH 7 on an ice-bath for another hour. The title compound was filtered off, washed with cold water, and dried under reduced pressure to yield a white solid (11.262 g, 23.7 mmol, 88%).

Automated:

The Backbone was flushed with little ^tBuOK in ^tBuOH (0.5 M) to remove traces of water. The crude carboxamide (12.1365 g, 24.6 mmol) was transferred from the rotary evaporator flask to the reactor with one portion of ^tBuOK in ^tBuOH (53.5 mL, 0.5 M, 1.1 eq.) at 60°C and held at reflux (105°C) for 8 hours. The dead volume beneath the sinter plate was filled with 25 mL of water. After cooling to 10°C the solution was quenched with water (55.5 mL) and transferred to the filter flask. At 10°C, a dilute solution of HCl (54.83 mL, 0.447 M, 1 eq.) was added dropwise (0.5 mL/min). The resulting suspension was stirred for 60 min, filtered off, washed twice with water (10 mL) and dried under a stream of argon (6 hours, 50°C). A yellowish powder was obtained (8.2 g, 70%, 46% overall yield).

¹H NMR (600 MHz, CDCl₃): δ 10.91 (s(br) 1H), 8.83 (d, *J* = 2.4 Hz, 1H), 7.84 (dd, *J* = 8.7 Hz, 2.4 Hz, 1H), 7.15 (d, *J* = 8.8 Hz, 1H), 4.37 (q, *J* = 7.0 Hz, 1H), 4.27 (s, 3H), 3.11 (s(br), 4H), 2.93 (t, *J* = 7.6 Hz, 2H), 2.50 (m (br), 4H), 2.27 (s, 3H), 1.86 (hex, *J* = 7.4 Hz, 2H), 1.65 (t, *J* = 7.0 Hz, 4H), 1.02 (t, *J* = 7.4 Hz, 3H).

¹³C NMR (151 MHz, CDCl₃): δ 159.4, 153.8, 147.2, 146.5, 138.5, 131.8, 131.3, 129.2, 124.7, 121.2, 113.2, 66.2, 54.2, 46.1, 45.9, 38.4, 27.9, 22.4, 14.7, 14.2.

HRESI-MS: [C₂₂H₃₀N₆O₄S+H]⁺ calculated 475.2122 found 475.2112.

EXPERIMENTAL

REFERENCES

1. Ley, S. V.; Fitzpatrick, D. E.; Ingham, R. J.; Myers, R. M., Organic synthesis: march of the machines. *Angew Chem Int Ed Engl* **2015**, *54* (11), 3449-64.
2. Merrifield, R. B., Automated synthesis of peptides. *Science* **1965**, *150* (3693), 178-85.
3. Alvarado-Urbina, G.; Sathe, G. M.; Liu, W. C.; Gillen, M. F.; Duck, P. D.; Bender, R.; Ogilvie, K. K., Automated synthesis of gene fragments. *Science* **1981**, *214* (4518), 270-4.
4. Plante, O. J.; Palmacci, E. R.; Seeberger, P. H., Automated solid-phase synthesis of oligosaccharides. *Science* **2001**, *291* (5508), 1523-7.
5. Li, J.; Ballmer, S. G.; Gillis, E. P.; Fujii, S.; Schmidt, M. J.; Palazzolo, A. M.; Lehmann, J. W.; Morehouse, G. F.; Burke, M. D., Synthesis of many different types of organic small molecules using one automated process. *Science* **2015**, *347* (6227), 1221-6.
6. Merrifield, R. B., Solid Phase Peptide Synthesis. I. The Synthesis of a Tetrapeptide. *Journal of the American Chemical Society* **1963**, *85* (14), 2149-2154.
7. Meienhofer, J.; Schnabel, E.; Bremer, H.; Brinkhoff, O.; Zabel, R.; Sroka, W.; Klostermeyer, H.; Brandenburg, D.; Okuda, T.; Zahn, H., Synthesis of the insulin chain and the combination to insulin active preparation. *Z. Naturforsch.* **1963**, *18b* (12), 1120-1.
8. Katsoyannis, P. G.; Tometsko, A.; Fukuda, K., Insulin Peptides. IX. The Synthesis of the A-Chain of Insulin and its Combination with Natural B-Chain to Generate Insulin Activity. *Journal of the American Chemical Society* **1963**, *85* (18), 2863-2865.
9. Sun, Y., The creation of synthetic crystalline bovine insulin. *Protein Cell* **2015**, *6* (11), 781-83.
10. Katsoyannis, P. G., Insulin Peptides. I. Synthesis of Cysteine-Containing Peptides Related to the A-Chain of Sheep Insulin. *Journal of the American Chemical Society* **1961**, *83* (19), 4053-4057.
11. Katsoyannis, P. G., Synthesis of Insulin. *Science* **1966**, *154* (3756), 1509-1514.
12. Gutte, B.; Merrifield, R. B., The total synthesis of an enzyme with ribonuclease A activity. *J Am Chem Soc* **1969**, *91* (2), 501-2.
13. Merrifield, R. B., The chemical synthesis of proteins. *Protein Sci* **1996**, *5* (9), 1947-51.
14. Jaradat, D. M. M., Thirteen decades of peptide synthesis: key developments in solid phase peptide synthesis and amide bond formation utilized in peptide ligation. *Amino Acids* **2018**, *50* (1), 39-68.
15. Michelson, A. M.; Todd, A. R., Nucleotides part XXXII. Synthesis of a dithymidine dinucleotide containing a 3': 5'-internucleotidic linkage. *J. Chem. Soc.* **1955**, *0* (0), 2632-2638.
16. Letsinger, R. L.; Mahadevan, V., Stepwise Synthesis of Oligodeoxyribonucleotides on an Insoluble Polymer Support. *Journal of the American Chemical Society* **1966**, *88* (22), 5319-5324.
17. Letsinger, R. L.; Ogilvie, K. K., Convenient method for stepwise synthesis of oligothymidylate derivatives in large-scale quantities. *Journal of the American Chemical Society* **1967**, *89* (18), 4801-4803.

REFERENCES

18. Khorana, H. G., *Some recent developments in the chemistry of phosphate esters of biological interest*. Wiley: 1961.
19. Letsinger, R. L.; Finnan, J. L.; Heavner, G. A.; Lunsford, W. B., Nucleotide chemistry. XX. Phosphite coupling procedure for generating internucleotide links. *Journal of the American Chemical Society* **1975**, *97* (11), 3278-3279.
20. Ogilvie, K. K.; Nemer, M. J., Silica-Gel as Solid Support in the Synthesis of Oligoribonucleotides. *Tetrahedron Lett* **1980**, *21* (43), 4159-4162.
21. Matteucci, M. D.; Caruthers, M. H., The synthesis of oligodeoxyrimidines on a polymer support. *Tetrahedron Lett* **1980**, *21* (8), 719-722.
22. Kosuri, S.; Church, G. M., Large-scale de novo DNA synthesis: technologies and applications. *Nat Methods* **2014**, *11* (5), 499-507.
23. Seeberger, P. H.; Werz, D. B., Automated synthesis of oligosaccharides as a basis for drug discovery. *Nat Rev Drug Discov* **2005**, *4* (9), 751-63.
24. Schuerch, C.; Frechet, J. M., Solid-phase synthesis of oligosaccharides. I. Preparation of the solid support. Poly[p-(1-propen-3-ol-1-yl)styrene]. *Journal of the American Chemical Society* **1971**, *93* (2), 492-496.
25. Danishefsky, S. J.; McClure, K. F.; Randolph, J. T.; Ruggeri, R. B., A strategy for the solid-phase synthesis of oligosaccharides. *Science* **1993**, *260* (5112), 1307-9.
26. Seeberger, P. H.; Haase, W. C., Solid-phase oligosaccharide synthesis and combinatorial carbohydrate libraries. *Chem Rev* **2000**, *100* (12), 4349-94.
27. Liang, R.; Yan, L.; Loebach, J.; Ge, M.; Uozumi, Y.; Sekanina, K.; Horan, N.; Gildersleeve, J.; Thompson, C.; Smith, A.; Biswas, K.; Still, W. C.; Kahne, D., Parallel synthesis and screening of a solid phase carbohydrate library. *Science* **1996**, *274* (5292), 1520-2.
28. Seeberger, P. H., Automated carbohydrate synthesis to drive chemical glycomics. *Chem Commun (Camb)* **2003**, (10), 1115-21.
29. Naresh, K.; Schumacher, F.; Hahm, H. S.; Seeberger, P. H., Pushing the limits of automated glycan assembly: synthesis of a 50mer polymannoside. *Chem Commun (Camb)* **2017**, *53* (65), 9085-9088.
30. Gillis, E. P.; Burke, M. D., A simple and modular strategy for small molecule synthesis: iterative Suzuki-Miyaura coupling of B-protected haloboronic acid building blocks. *Journal of the American Chemical Society* **2007**, *129* (21), 6716-7.
31. Woerly, E. M.; Roy, J.; Burke, M. D., Synthesis of most polyene natural product motifs using just 12 building blocks and one coupling reaction. *Nature Chemistry* **2014**, *6*, 484.
32. Service, R., Billion-dollar project would synthesize hundreds of thousands of molecules in search of new medicines. *Science* **2017**.
33. Lehmann, J. W.; Blair, D. J.; Burke, M. D., Toward Generalization of Iterative Small Molecule Synthesis. *Nat Rev Chem* **2018**, *2* (2).
34. Dessy, R., Robots in the Laboratory: Part I. *Analytical Chemistry* **1983**, *55* (11), 1100A-1114A.
35. Frisbee, A. R.; Nantz, M. H.; Kramer, G. W.; Fuchs, P. L., Robotic orchestration of organic reactions: yield optimization via an automated system with operator-specified reaction sequences. *Journal of the American Chemical Society* **1984**, *106* (23), 7143-7145.

REFERENCES

36. Emiabata-Smith, D. F.; Crookes, D. L.; Owen, M. R., A practical approach to accelerated process screening and optimisation. *Organic Process Research & Development* **1999**, *3* (4), 281-288.
37. Kenny, B. A.; Bushfield, M.; Parry-Smith, D. J.; Fogarty, S.; Treherne, J. M., The application of high-throughput screening to novel lead discovery. In *Progress in Drug Research*, Jucker, E., Ed. Birkhäuser Basel: Basel, 1998; pp 245-269.
38. Mellor, G. W.; Fogarty, S. J.; O'Brien, M. S.; Congreve, M.; Banks, M. N.; Mills, K. M.; Jefferies, B.; Houston, J. G., Searching for chemokine receptor binding antagonists by high throughput screening. *Journal of Biomolecular Screening* **1997**, *2* (3), 153-157.
39. Szostak, J. W., Introduction: Combinatorial Chemistry. *Chem Rev* **1997**, *97* (2), 347-348.
40. DeWitt, S. H.; Czarnik, A. W., Automated synthesis and combinatorial chemistry. *Curr Opin Biotechnol* **1995**, *6* (6), 640-5.
41. Cargill, J. F.; Lebl, M., New methods in combinatorial chemistry - robotics and parallel synthesis. *Current Opinion in Chemical Biology* **1997**, *1* (1), 67-71.
42. Godfrey, A. G.; Masquelin, T.; Hemmerle, H., A remote-controlled adaptive medchem lab: an innovative approach to enable drug discovery in the 21st Century. *Drug Discov Today* **2013**, *18* (17-18), 795-802.
43. Rubin, A. E.; Tummala, S.; Both, D. A.; Wang, C.; Delaney, E. J., Emerging technologies supporting chemical process R&D and their increasing impact on productivity in the pharmaceutical industry. *Chem Rev* **2006**, *106* (7), 2794-810.
44. Potyrailo, R.; Rajan, K.; Stoewe, K.; Takeuchi, I.; Chisholm, B.; Lam, H., Combinatorial and high-throughput screening of materials libraries: review of state of the art. *ACS Comb Sci* **2011**, *13* (6), 579-633.
45. Chemspeed Technologies. <http://www.chemspeed.com/> (accessed 23/03/2017).
46. TECAN Group. <http://www.tecan.com/> (accessed 22/03/2017).
47. Zinsser Analytic. <http://www.zinsser-analytic.com/> (accessed 23/03/2017).
48. Schneider, G., Automating drug discovery. *Nat Rev Drug Discov* **2017**.
49. Vickerstaffe, E.; Warrington, B. H.; Ladlow, M.; Ley, S. V., Fully automated multi-step solution phase synthesis using polymer supported reagents: preparation of histone deacetylase inhibitors. *Organic & Biomolecular Chemistry* **2003**, *1* (14), 2419-22.
50. Bartlett, P. A.; Entzeroth, M., The Use of Polymer-Assisted Solution-Phase Synthesis and Automation for the High-Throughput Preparation of Biologically Active Compounds. In *Exploiting Chemical Diversity for Drug Discovery*, 2006; pp 1-32.
51. Kirschning, A.; Monenschein, H.; Wittenberg, R., Functionalized Polymers- Emerging Versatile Tools for Solution-Phase Chemistry and Automated Parallel Synthesis. *Angew Chem Int Ed Engl* **2001**, *40* (4), 650-679.
52. Van Loo, M. E.; Lengowski, P. E., Automated Workstations for Parallel Synthesis. *Organic Process Research & Development* **2002**, *6* (6), 833-840.
53. Hartman, R. L.; McMullen, J. P.; Jensen, K. F., Deciding whether to go with the flow: evaluating the merits of flow reactors for synthesis. *Angew Chem Int Ed Engl* **2011**, *50* (33), 7502-19.

REFERENCES

54. Kockmann, N.; Thenee, P.; Fleischer-Trebes, C.; Laudadio, G.; Noel, T., Safety assessment in development and operation of modular continuous-flow processes. *React Chem Eng* **2017**, *2* (3), 258-280.
55. Hartman, R. L., Managing Solids in Microreactors for the Upstream Continuous Processing of Fine Chemicals. *Organic Process Research & Development* **2012**, *16* (5), 870-887.
56. Plouffe, P.; Macchi, A.; Roberge, D. M., From Batch to Continuous Chemical Synthesis—A Toolbox Approach. *Organic Process Research & Development* **2014**, *18* (11), 1286-1294.
57. Plutschack, M. B.; Pieber, B.; Gilmore, K.; Seeberger, P. H., The Hitchhiker's Guide to Flow Chemistry parallel. *Chem Rev* **2017**, *117* (18), 11796-11893.
58. Roberge, D. M.; Zimmermann, B.; Rainone, F.; Gottspöner, M.; Eyholzer, M.; Kockmann, N., Microreactor Technology and Continuous Processes in the Fine Chemical and Pharmaceutical Industry: Is the Revolution Underway? *Organic Process Research & Development* **2008**, *12* (5), 905-910.
59. Valera, F. E.; Quaranta, M.; Moran, A.; Blacker, J.; Armstrong, A.; Cabral, J. T.; Blackmond, D. G., The flow's the thing..or is it? Assessing the merits of homogeneous reactions in flask and flow. *Angew Chem Int Ed Engl* **2010**, *49* (14), 2478-85.
60. McQuade, D. T.; Seeberger, P. H., Applying flow chemistry: methods, materials, and multistep synthesis. *J Org Chem* **2013**, *78* (13), 6384-9.
61. Chandra, T.; Zebrowski, J. P., Hazards associated with laboratory scale hydrogenations. *Journal of Chemical Health and Safety* **2016**, *23* (4), 16-25.
62. Jones, R. V.; Godorhazy, L.; Varga, N.; Szalay, D.; Urge, L.; Darvas, F., Continuous-flow high pressure hydrogenation reactor for optimization and high-throughput synthesis. *J Comb Chem* **2006**, *8* (1), 110-6.
63. Dormán, G.; Kocsis, L.; Jones, R.; Darvas, F., A benchtop continuous flow reactor: A solution to the hazards posed by gas cylinder based hydrogenation. *Journal of Chemical Health and Safety* **2013**, *20* (4), 3-8.
64. Maas, G., New syntheses of diazo compounds. *Angew Chem Int Ed Engl* **2009**, *48* (44), 8186-95.
65. Black, T. H., The preparation and reactions of diazomethane. *Aldrichimica Acta* **1983**, *16* (1), 3-10.
66. Lewis, C. E., Diazomethane Poisoning. Report of a Case Suggesting Sensitization Reaction. *J Occup Med* **1964**, *6*, 91-2.
67. LeWinn, E. B., Diazomethane poisoning; report of a fatal case with autopsy. *Am J Med Sci* **1949**, *218* (5), 556-62.
68. Schoental, R., Carcinogenic action of diazomethane and of nitroso-n-methyl urethane. *Nature* **1960**, *188*, 420-1.
69. de Boer, T. J.; Backer, H. J., Diazomethane. *Organic Syntheses* **1956**, *36*, 16-19.
70. Sigma-Aldrich. Diazald® and Diazomethane Generators *Aldrich Technical Bulletins* [Online], 2004. https://www.sigmaaldrich.com/content/dam/sigma-aldrich/docs/Aldrich/Bulletin/al_techbull_al180.pdf (accessed 18/07/2018).
71. Maurya, R. A.; Park, C. P.; Lee, J. H.; Kim, D. P., Continuous in situ generation, separation, and reaction of diazomethane in a dual-channel microreactor. *Angew Chem Int Ed Engl* **2011**, *50* (26), 5952-5.

REFERENCES

72. Mastronardi, F.; Gutmann, B.; Kappe, C. O., Continuous flow generation and reactions of anhydrous diazomethane using a Teflon AF-2400 tube-in-tube reactor. *Org Lett* **2013**, *15* (21), 5590-3.
73. O'Brien, M.; Baxendale, I. R.; Ley, S. V., Flow ozonolysis using a semipermeable Teflon AF-2400 membrane to effect gas-liquid contact. *Org Lett* **2010**, *12* (7), 1596-8.
74. Acke, D. R. J.; Stevens, C. V., A HCN-based reaction under microreactor conditions: industrially feasible and continuous synthesis of 3,4-diamino-1H-isochromen-1-ones. *Green Chemistry* **2007**, *9* (4), 386-390.
75. Newman, S. G.; Jensen, K. F., The role of flow in green chemistry and engineering. *Green Chemistry* **2013**, *15* (6), 1456-1472.
76. Tilstam, U., A Continuous Methylation of Phenols and N,H-Heteroaromatic Compounds with Dimethyl Carbonate. *Organic Process Research & Development* **2012**, *16* (12), 1974-1978.
77. Adeyemi, A.; Bergman, J.; Branalt, J.; Savmarker, J.; Larhed, M., Continuous Flow Synthesis under High-Temperature/High-Pressure Conditions Using a Resistively Heated Flow Reactor. *Organic Process Research & Development* **2017**, *21* (7), 947-955.
78. Hamley, P. A.; Ilkenhans, T.; Webster, J. M.; Garcia-Verdugo, E.; Venardou, E.; Clarke, M. J.; Auerbach, R.; Thomas, W. B.; Whiston, K.; Poliakoff, M., Selective partial oxidation in supercritical water: the continuous generation of terephthalic acid from para-xylene in high yield. *Green Chemistry* **2002**, *4* (3), 235-238.
79. Banister, J. A.; Lee, P. D.; Poliakoff, M., Flow Reactors for Preparative Chemistry in Supercritical Fluid Solution: "Solvent-Free" Synthesis and Isolation of Cr(CO)₅(C₂H₄) and (η⁵-C₅H₅)Mn(CO)₂(η²-H₂). *Organometallics* **1995**, *14* (8), 3876-3885.
80. Han, X.; Poliakoff, M., Continuous reactions in supercritical carbon dioxide: problems, solutions and possible ways forward. *Chem Soc Rev* **2012**, *41* (4), 1428-36.
81. Licence, P.; Ke, J.; Sokolova, M.; Ross, S. K.; Poliakoff, M., Chemical reactions in supercritical carbon dioxide: from laboratory to commercial plant. *Green Chemistry* **2003**, *5* (2), 99-104.
82. Newton, S.; Carter, C. F.; Pearson, C. M.; de C. Alves, L.; Lange, H.; Thansandote, P.; Ley, S. V., Accelerating spirocyclic polyketide synthesis using flow chemistry. *Angew Chem Int Ed Engl* **2014**, *53* (19), 4915-20.
83. Lin, H.; Dai, C.; Jamison, T. F.; Jensen, K. F., A Rapid Total Synthesis of Ciprofloxacin Hydrochloride in Continuous Flow. *Angew Chem Int Ed Engl* **2017**, *56* (30), 8870-8873.
84. Fitzpatrick, D. E.; Ley, S. V., Engineering chemistry: integrating batch and flow reactions on a single, automated reactor platform. *React Chem Eng* **2016**, *1* (6), 629-635.
85. Bana, P.; Orkenyi, R.; Lovei, K.; Lako, A.; Turos, G. I.; Eles, J.; Faigl, F.; Greiner, I., The route from problem to solution in multistep continuous flow synthesis of pharmaceutical compounds. *Bioorg Med Chem* **2016**.
86. Porta, R.; Benaglia, M.; Puglisi, A., Flow Chemistry: Recent Developments in the Synthesis of Pharmaceutical Products. *Organic Process Research & Development* **2015**, *20* (1), 2-25.

REFERENCES

87. Laue, S.; Haverkamp, V.; Mleczko, L., Experience with Scale-Up of Low-Temperature Organometallic Reactions in Continuous Flow. *Organic Process Research & Development* **2016**, *20* (2), 480-486.
88. Styring, P.; Parracho, A. I., From discovery to production: scale-out of continuous flow meso reactors. *Beilstein J Org Chem* **2009**, *5*, 29.
89. Monbaliu, J. C. M.; Stelzer, T.; Revalor, E.; Weeranoppanant, N.; Jensen, K. F.; Myerson, A. S., Compact and Integrated Approach for Advanced End-to-End Production, Purification, and Aqueous Formulation of Lidocaine Hydrochloride. *Organic Process Research & Development* **2016**, *20* (7), 1347-1353.
90. Mascia, S.; Heider, P. L.; Zhang, H.; Lakerveld, R.; Benyahia, B.; Barton, P. I.; Braatz, R. D.; Cooney, C. L.; Evans, J. M.; Jamison, T. F.; Jensen, K. F.; Myerson, A. S.; Trout, B. L., End-to-end continuous manufacturing of pharmaceuticals: integrated synthesis, purification, and final dosage formation. *Angew Chem Int Ed Engl* **2013**, *52* (47), 12359-63.
91. Adamo, A.; Beingessner, R. L.; Behnam, M.; Chen, J.; Jamison, T. F.; Jensen, K. F.; Monbaliu, J. C.; Myerson, A. S.; Revalor, E. M.; Snead, D. R.; Stelzer, T.; Weeranoppanant, N.; Wong, S. Y.; Zhang, P., On-demand continuous-flow production of pharmaceuticals in a compact, reconfigurable system. *Science* **2016**, *352* (6281), 61-7.
92. Deming, S. N.; Pardue, H. L., Automated instrumental system for fundamental characterization of chemical reactions. *Analytical Chemistry* **1971**, *43* (2), 192-200.
93. Winicov, H.; Schainbaum, J.; Buckley, J.; Longino, G.; Hill, J.; Berkoff, C. E., Chemical process optimization by computer — a self-directed chemical synthesis system. *Analytica Chimica Acta* **1978**, *103* (4), 469-476.
94. Chodosh, D. F.; Wdzieckowski, F. E.; Schainbaum, J.; Berkoff, C. E., Automated chemical synthesis. Part 2: Interfacing strategies. *J Automat Chem* **1983**, *5* (2), 99-102.
95. Chodosh, D. F.; Levinson, S. H.; Weber, J. L.; Kamholz, K.; Berkoff, C. E., Automated chemical synthesis. Part 3: Temperature control systems. *J Automat Chem* **1983**, *5* (2), 103-7.
96. Chodosh, D. F.; Kamholz, K.; Levinson, S. H.; Rhinesmith, R., Automated chemical synthesis. Part 4: Batch-type reactor automation and real-time software design. *J Automat Chem* **1986**, *8* (3), 106-21.
97. Legrand, M.; Foucard, A., Automation on the laboratory bench. *J Chem Educ* **1978**, *55* (12), 767.
98. Legrand, M.; Bolla, P., A fully automatic apparatus for chemical reactions on the laboratory scale. *J Automat Chem* **1985**, *7* (1), 31-7.
99. Porte, C.; Roussin, D.; Bondiou, J. C.; Hodac, F.; Delacroix, A., The 'Automated Versatile Modular Reactor': construction and use. *J Automat Chem* **1987**, *9* (4), 166-73.
100. Porte, C.; Hamdan, F.; Delacroix, A., Automated medium-pressure modular liquid-gas reactor. *J Automat Chem* **1989**, *11* (4), 168-73.
101. Hayashi, N.; Sugawara, T.; Shintani, M.; Kato, S., Computer-assisted automatic synthesis II. Development of a fully automated apparatus for preparing substituted N-(carboxyalkyl)amino acids. *J Automat Chem* **1989**, *11* (5), 212-20.
102. Sugawara, T.; Kato, S.; Okamoto, S., Development of fully-automated synthesis systems. *J Automat Chem* **1994**, *16* (1), 33-42.

REFERENCES

103. Machida, K.; Hirose, Y.; Fuse, S.; Sugawara, T.; Takahashi, T., Development and application of a solution-phase automated synthesizer, 'ChemKonzert'. *Chem Pharm Bull (Tokyo)* **2010**, *58* (1), 87-93.
104. Doi, T.; Fuse, S.; Miyamoto, S.; Nakai, K.; Sasuga, D.; Takahashi, T., A formal total synthesis of taxol aided by an automated synthesizer. *Chem Asian J* **2006**, *1* (3), 370-83.
105. Tanaka, Y.; Fuse, S.; Tanaka, H.; Doi, T.; Takahashi, T., An Efficient Synthesis of a Cyclic Ether Key Intermediate for 9-Membered Masked Eneidyne Using an Automated Synthesizer. *Organic Process Research & Development* **2009**, *13* (6), 1111-1121.
106. Fuse, S.; Okada, K.; Iijima, Y.; Munakata, A.; Machida, K.; Takahashi, T.; Takagi, M.; Shin-ya, K.; Doi, T., Total synthesis of spiruchostatin B aided by an automated synthesizer. *Org Biomol Chem* **2011**, *9* (10), 3825-33.
107. Fuse, S.; Machida, K.; Takahashi, T., Efficient Synthesis of Natural Products Aided by Automated Synthesizers and Microreactors. In *New Strategies in Chemical Synthesis and Catalysis*, 2012; pp 33-57.
108. Fuse, S.; Ikebe, A.; Oosumi, K.; Karasawa, T.; Matsumura, K.; Izumikawa, M.; Johmoto, K.; Uekusa, H.; Shin-ya, K.; Doi, T.; Takahashi, T., Asymmetric Total Synthesis of ent-Pyripyropene A. *Chemistry* **2015**, *21* (26), 9454-60.
109. Masui, H.; Yosugi, S.; Fuse, S.; Takahashi, T., Solution-phase automated synthesis of an alpha-amino aldehyde as a versatile intermediate. *Beilstein J Org Chem* **2017**, *13*, 106-110.
110. Orita, A.; Yasui, Y.; Otera, J., Automated synthesis: Development of a new apparatus friendly to synthetic chemists (MEDLEY). *Organic Process Research & Development* **2000**, *4* (5), 333-336.
111. MettlerToledo OptiMax.
https://www.mt.com/gb/en/home/products/L1_AutochemProducts/Chemical-Synthesis-and-Process-Development-Lab-Reactors/Synthesis-Reactor-Systems/optimax.html (accessed 19/07/2018).
112. Radleys Mya 4. <https://www.radleys.com/products/our-products/mya-4-reaction-station> (accessed 19/07/2018).
113. Syrris Atlas. <https://syrris.com/families/atlas-hd/> (accessed 19/07/2018).
114. Badamasi, Y. A. In *The working principle of an Arduino*, 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), Sept. 29 2014-Oct. 1 2014; 2014; pp 1-4.
115. Deadman, B. J.; Battilocchio, C.; Sliwinski, E.; Ley, S. V., A prototype device for evaporation in batch and flow chemical processes. *Green Chemistry* **2013**, *15* (8), 2050-2055.
116. Adamo, A.; Heider, P. L.; Weeranoppanant, N.; Jensen, K. F., Membrane-Based, Liquid-Liquid Separator with Integrated Pressure Control. *Industrial & Engineering Chemistry Research* **2013**, *52* (31), 10802-10808.
117. Dessimoz, A.-L.; Cavin, L.; Renken, A.; Kiwi-Minsker, L., Liquid-liquid two-phase flow patterns and mass transfer characteristics in rectangular glass microreactors. *Chemical Engineering Science* **2008**, *63* (16), 4035-4044.
118. Grzyb, J. A.; Batey, R. A., Achieving functional group diversity in parallel synthesis: solution-phase synthesis of a library of ureas, carbamates,

REFERENCES

- thiocarbamates, and amides using carbamoylimidazolium salts. *Tetrahedron Lett* **2008**, *49* (36), 5279-5282.
119. O'Brien, M.; Koos, P.; Browne, D. L.; Ley, S. V., A prototype continuous-flow liquid-liquid extraction system using open-source technology. *Org Biomol Chem* **2012**, *10* (35), 7031-6.
120. Eppel, S.; Kachman, T. Computer vision-based recognition of liquid surfaces and phase boundaries in transparent vessels, with emphasis on chemistry applications *arXiv.org e-Print archive* [Online], 2014. <http://arxiv.org/abs/1404.7174>.
121. Krizhevsky, A.; Sutskever, I.; Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the Acm* **2017**, *60* (6), 84-90.
122. Krizhevsky, A.; Sutskever, I.; Hinton, G. E. In *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 2012; pp 1097-1105.
123. Terzic, E.; Terzic, J.; Nagarajah, R.; Alamgir, M., Capacitive Sensing Technology. In *A Neural Network Approach to Fluid Quantity Measurement in Dynamic Environments*, 2012; pp 11-37.
124. Weast, R. C.; Astle, M. J.; Beyer, W. H., *CRC Handbook of Chemistry and Physics: A Ready-reference Book of Chemical and Physical Data*. CRC Press: 1983.
125. Microchip Technology Inc. Atmel Studio 7. <http://www.microchip.com/mplab/avr-support/atmel-studio-7> (accessed 17/08/2018).
126. Open Sound Control. <http://opensoundcontrol.org/osc> (accessed 20/08/2018).
127. McCurry, M. Simple Interpreter - C. <http://fundamental-code.com/interp/> (accessed 20/08/2018).
128. NAMUR Interessengemeinschaft Automatisierungstechnik der Prozessindustrie, Ausführung von elektrischen Steckverbindungen für die analoge und digitale Signalübertragung an Labor-MSR-Einzelgeräten. 1992; Vol. NE 28.
129. Raymond, E. S., *The new hacker's dictionary (3rd ed.)*. MIT Press: 1996; p 547.
130. Liechti, C. pySerial. <https://pythonhosted.org/pyserial/> (accessed 16/08/2018).
131. Erlang. <https://www.erlang.org/> (accessed 13/08/2018).
132. xacro. <http://wiki.ros.org/xacro> (accessed 13/08/2018).
133. NetworkX. <http://networkx.github.io/> (accessed 14/08/2018).
134. Raccuglia, P.; Elbert, K. C.; Adler, P. D. F.; Falk, C.; Wenny, M. B.; Mollo, A.; Zeller, M.; Friedler, S. A.; Schrier, J.; Norquist, A. J., Machine-learning-assisted materials discovery using failed experiments. *Nature* **2016**, *533* (7601), 73-7.
135. Brandes, U.; Eiglsperger, M.; Kaufmann, M.; Lerner, J.; Pich, P. The GraphML file format. <http://graphml.graphdrawing.org/index.html>.
136. yWorks yEd Graph Editor. <https://www.yworks.com/products/yed> (accessed 16/08/2018).
137. Backus, J. W.; Wegstein, J. H.; van Wijngaarden, A.; Woodger, M.; Bauer, F. L.; Green, J.; Katz, C.; McCarthy, J.; Perlis, A. J.; Rutishauser, H.; Samelson, K.; Vauquois, B., Report on the algorithmic language ALGOL 60. *Communications of the ACM* **1960**, *3* (5), 299-314.
138. Beazley, D. PLY (Python Lex-Yacc). <http://www.dabeaz.com/ply/> (accessed 15/08/2018).

REFERENCES

139. OpenCV. <https://opencv.org/> (accessed 15/08/2018).
140. Ahmadi, A.; Khalili, M.; Hajikhani, R.; Safari, N.; Nahri-Niknafs, B., Anti-inflammatory effects of two new methyl and morpholine derivatives of diphenhydramine on rats. *Medicinal Chemistry Research* **2011**, *21* (11), 3532-3540.
141. Dale, D. J.; Dunn, P. J.; Golightly, C.; Hughes, M. L.; Levett, P. C.; Pearce, A. K.; Searle, P. M.; Ward, G.; Wood, A. S., The chemical development of the commercial route to sildenafil: A case history. *Organic Process Research & Development* **2000**, *4* (1), 17-22.
142. Rieveschl, G., Jr. Dialkylaminoalkyl benzhydryl ethers. US2421714, June 3, 1947, 1947.
143. Rieveschl, G., Jr. 2-Dimethylaminoethyl p-halobenzhydryl ethers and their salts. US2527963, October 31, 1950, 1950.
144. am Ende, D. J.; Clifford, P. J.; DeAntonis, D. M.; SantaMaria, C.; Brenek, S. J., Preparation of Grignard Reagents: FTIR and Calorimetric Investigation for Safe Scale-Up. *Organic Process Research & Development* **1999**, *3* (5), 319-329.
145. Yue, M. H.; Sharkey, J. J.; Leung, J. C., Relief Vent Sizing for a Grignard Reaction. *Journal of Loss Prevention in the Process Industries* **1994**, *7* (5), 413-418.
146. Tilstam, U.; Weinmann, H., Activation of Mg metal for safe formation of Grignard reagents on plant scale. *Organic Process Research & Development* **2002**, *6* (6), 906-910.
147. Lai, Y. H., Grignard-Reagents from Chemically Activated Magnesium. *Synthesis-Stuttgart* **1981**, *1981* (8), 585-604.
148. Williamson, K. L., Starting the Grignard Reaction. *J Chem Educ* **1988**, *65* (4), 376-376.
149. Tuulmets, A.; Kaubi, K.; Heinoja, K., Influence of Sonication on Grignard-Reagent Formation. *Ultrasonics Sonochemistry* **1995**, *2* (2), S75-S78.
150. Meier, R. Preparation of fluorinated phenylalkyltriazoles as anticonvulsants and pharmaceutical compositions containing them. EP199262A2, 1986.
151. Wheless, J. W.; Vazquez, B., Rufinamide: a novel broad-spectrum antiepileptic drug. *Epilepsy Curr* **2010**, *10* (1), 1-6.
152. Kankan, R. N.; Rao, D. R.; Birari, D. R. Process for the preparation of rufinamide. US8183269B2, 2012.
153. Bell, A. S.; Brown, D.; Terrett, N. K. Preparation of pyrazolo[4,3-d]pyrimidin-7-ones as cardiovascular agents. EP463756A1, 1992.
154. Ellis, P.; Terrett, N. K. Pyrazolopyrimidinones for the treatment of impotence. WO9428902A1, 1994.
155. Woodman, E. K.; Chaffey, J. G. K.; Hopes, P. A.; Hose, D. R. J.; Gilday, J. P., N,N'-Carbonyldiimidazole-Mediated Amide Coupling: Significant Rate Enhancement Achieved by Acid Catalysis with Imidazole·HCl. *Organic Process Research & Development* **2009**, *13* (1), 106-113.
156. Wang, Z.; Wang, Y.; Cheng, Q.; Zhao, X. Method for preparing sildenafil. CN104211705A, 2014.
157. Robert Sanger, C.; Raymond Riegel, E., Pyrosulfurylchlorid und Chlorsulfonsäure. *Zeitschrift für anorganische Chemie* **1912**, *76* (1), 79-128.
158. Horsley, L. H., Table of Azeotropes and Nonazeotropes. *Analytical Chemistry* **1947**, *19* (8), 508-600.

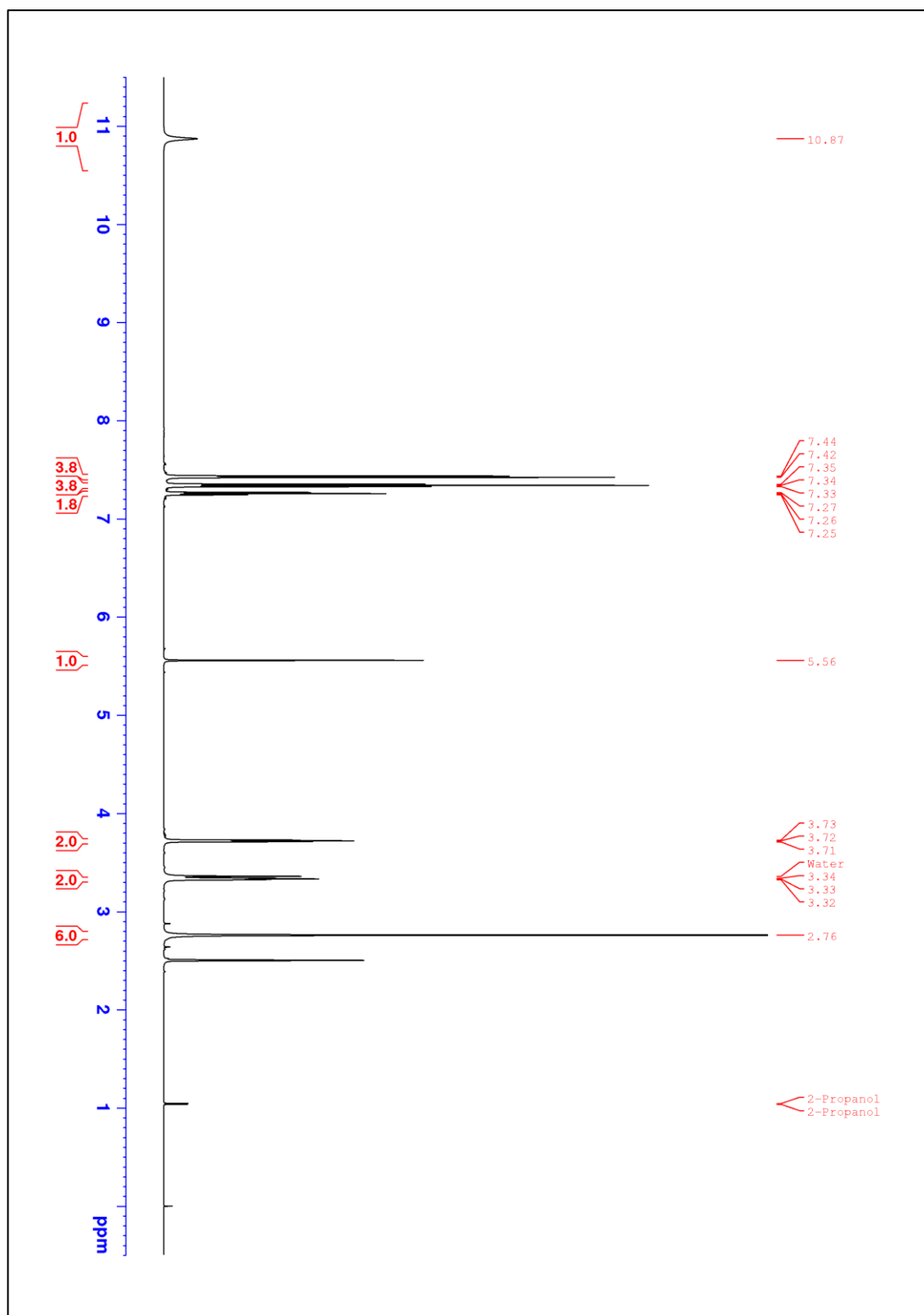
REFERENCES

159. Büchi Solution «Rotavapor® Dynamic Pro». <https://www.buchi.com/gb-en/products/laboratory-evaporation/solution-rotavapor-dynamic-pro> (accessed 24/08/2018).
160. Adusumilli, R.; Mallick, P., Data Conversion with ProteoWizard msConvert. In *Proteomics*, 2017; pp 339-368.

APPENDIX

I NMR SPECTRA

I.I DIPHENHYDRAMINE HYDROCHLORIDE (1)

Figure 93: ^1H NMR of diphenhydramine hydrochloride **1** produced manually

APPENDIX

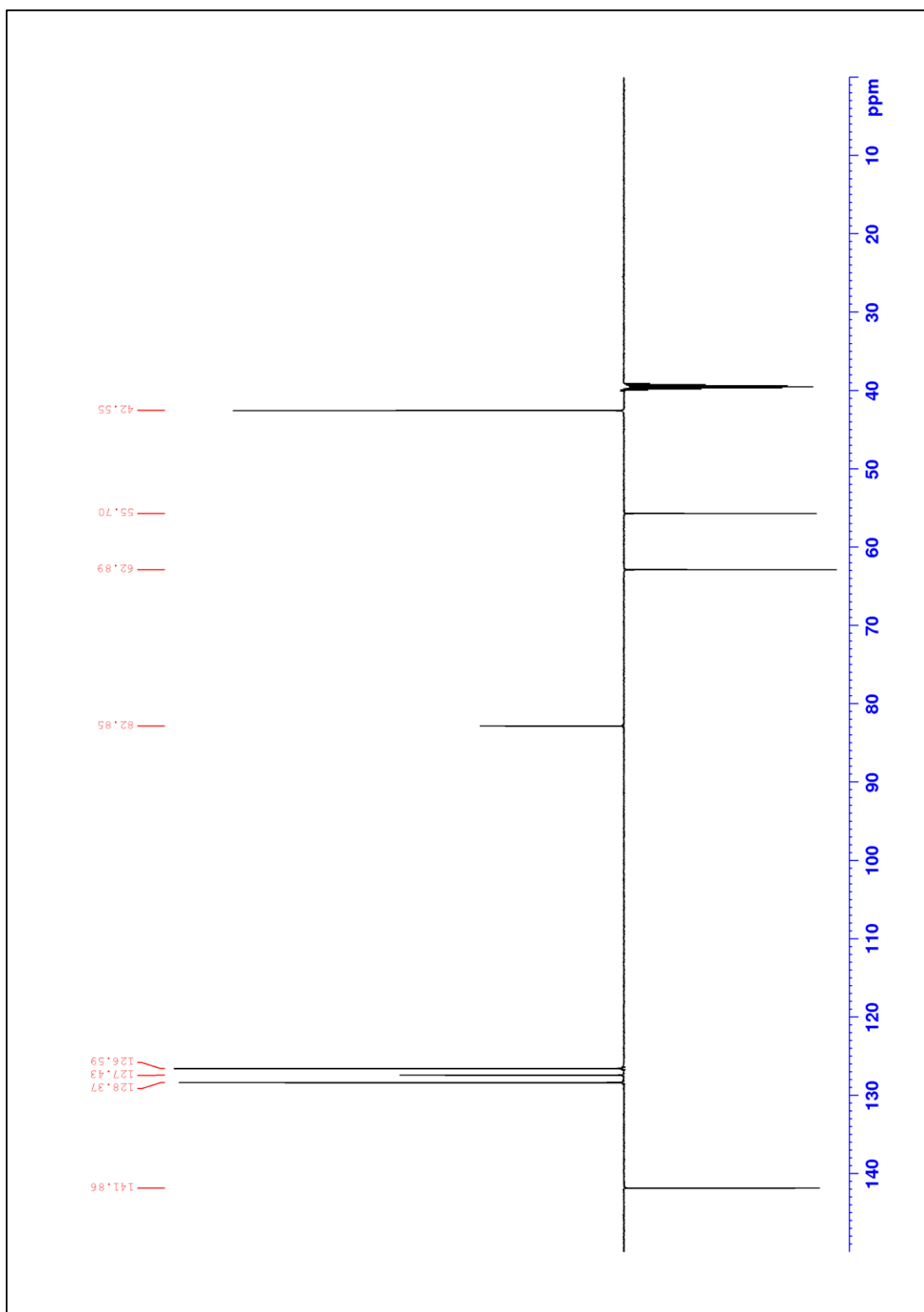


Figure 94: ^{13}C NMR (DEPTQ) of diphenhydramine hydrochloride **1** produced manually

APPENDIX

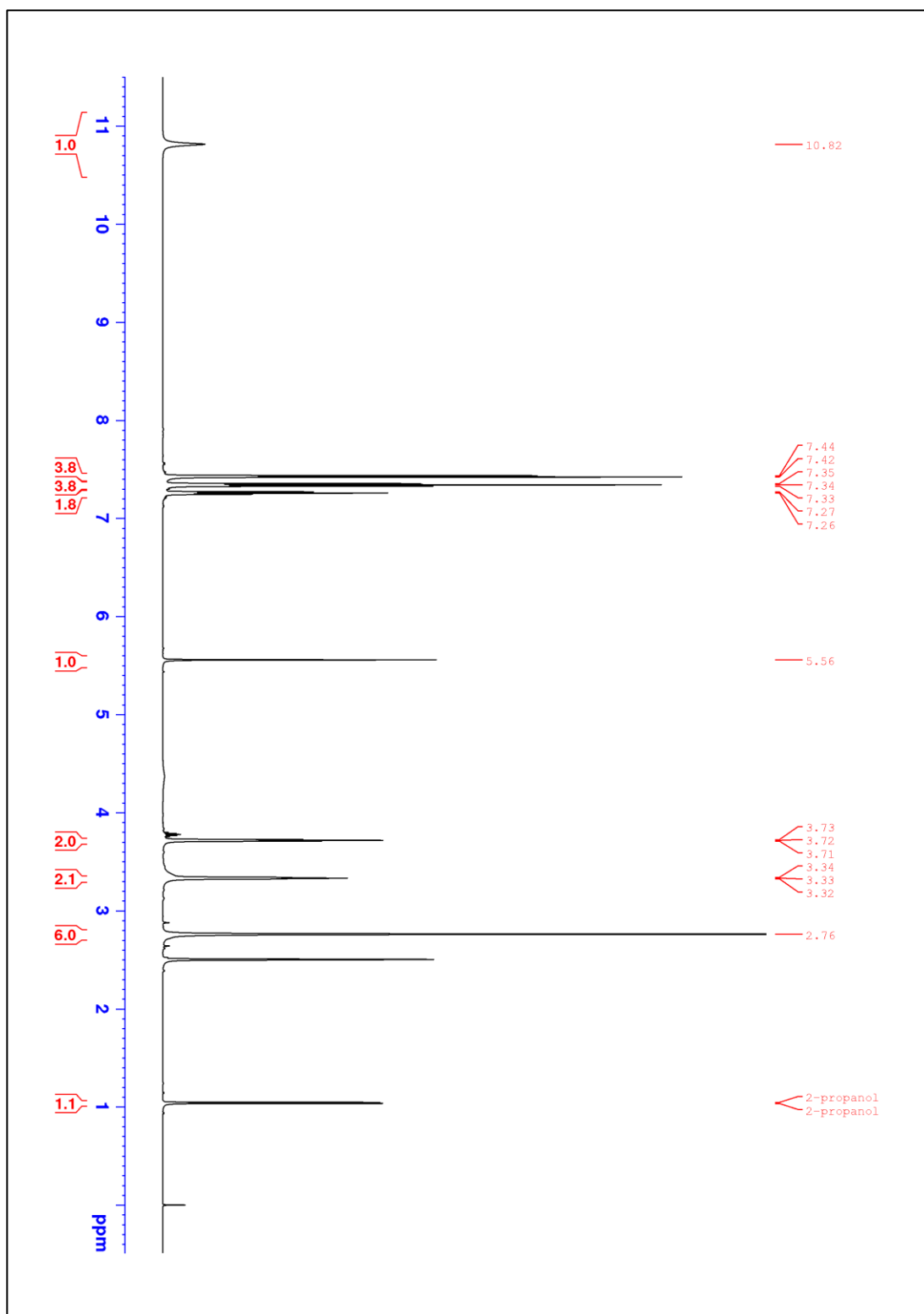


Figure 95: ^1H NMR of diphenhydramine hydrochloride **1** produced automatically

APPENDIX

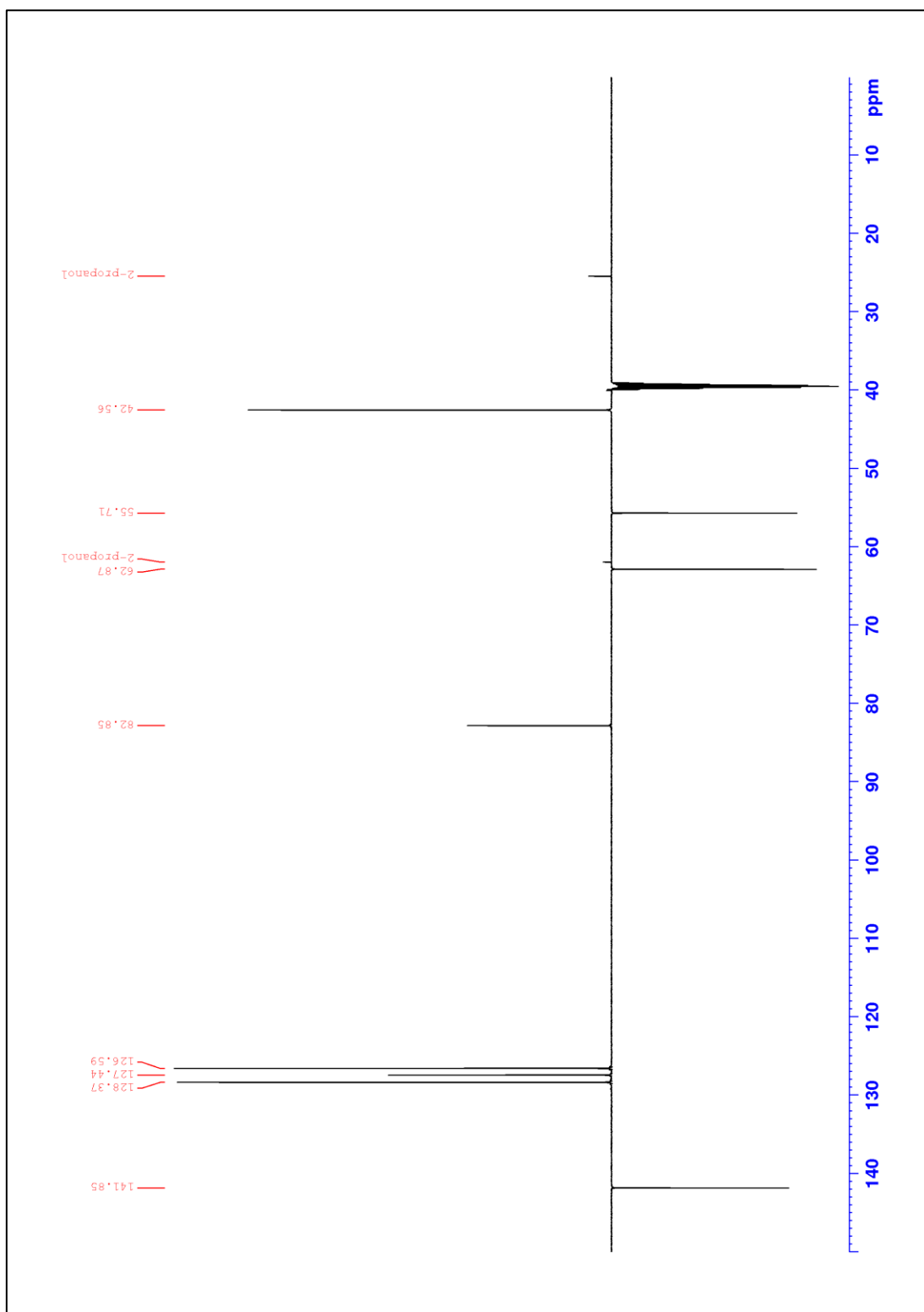


Figure 96: ¹³C NMR (DEPTQ) of diphenhydramine hydrochloride **1** produced automatically

APPENDIX

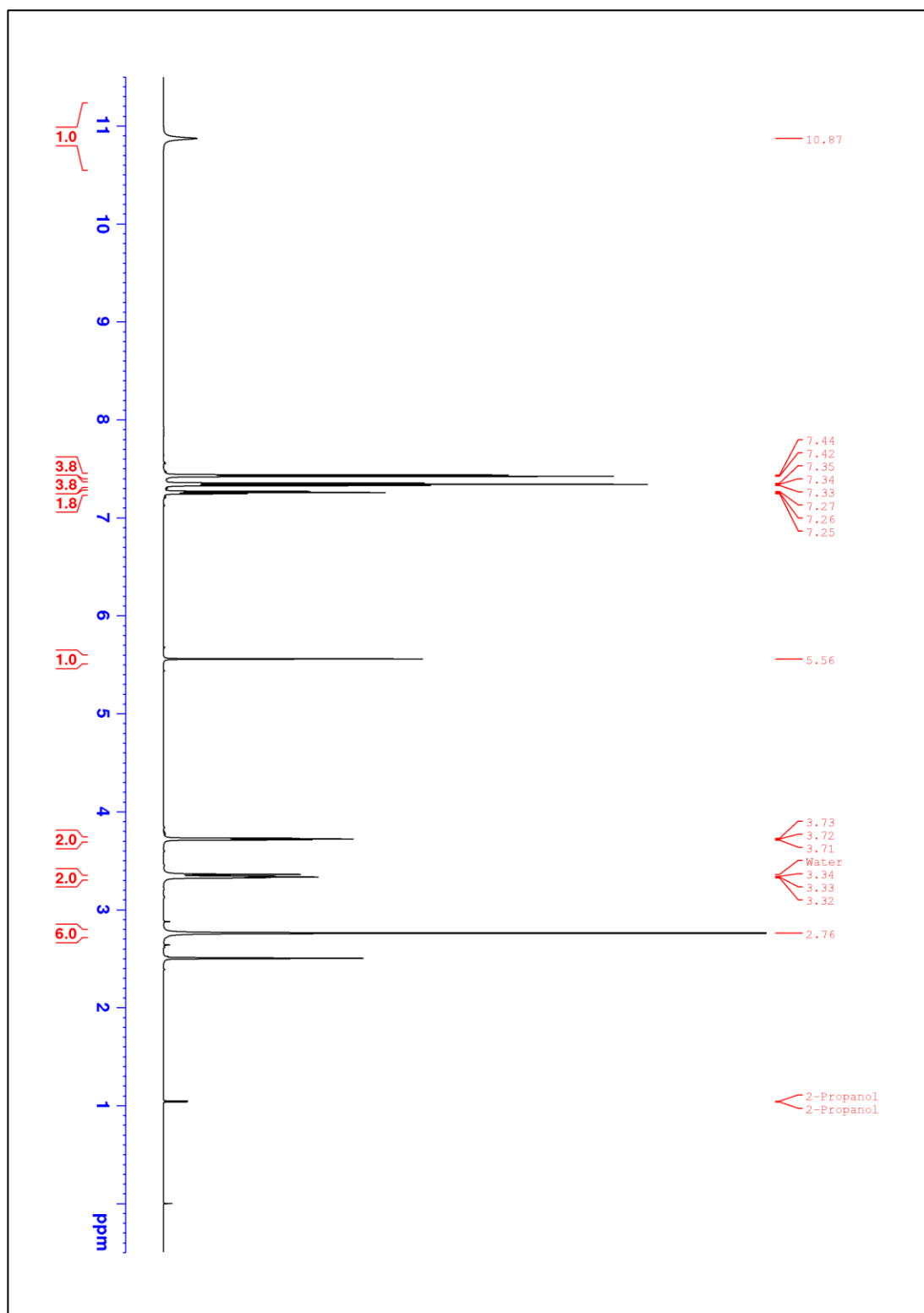


Figure 97: ^1H NMR of diphenhydramine hydrochloride **1** obtained from Sigma-Aldrich

APPENDIX

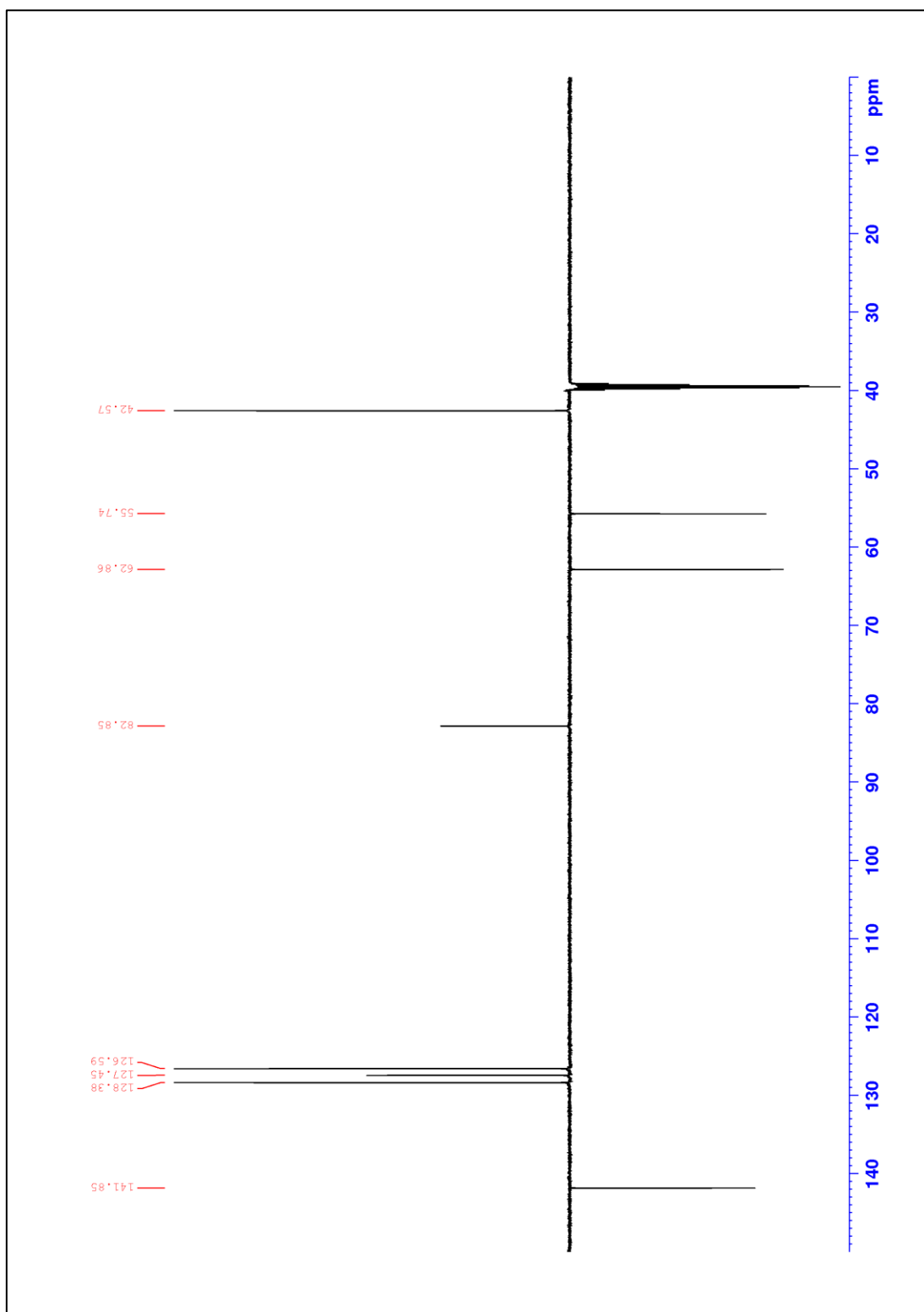


Figure 98: ¹³C NMR (DEPTQ) of diphenhydramine hydrochloride **1** obtained from Sigma-Aldrich

I.II 1-[(2,6-DIFLUOROPHENYL)METHYL]-1H-1,2,3-TRIAZOLE-4-CARBOXAMIDE
(RUFINAMIDE) (2)

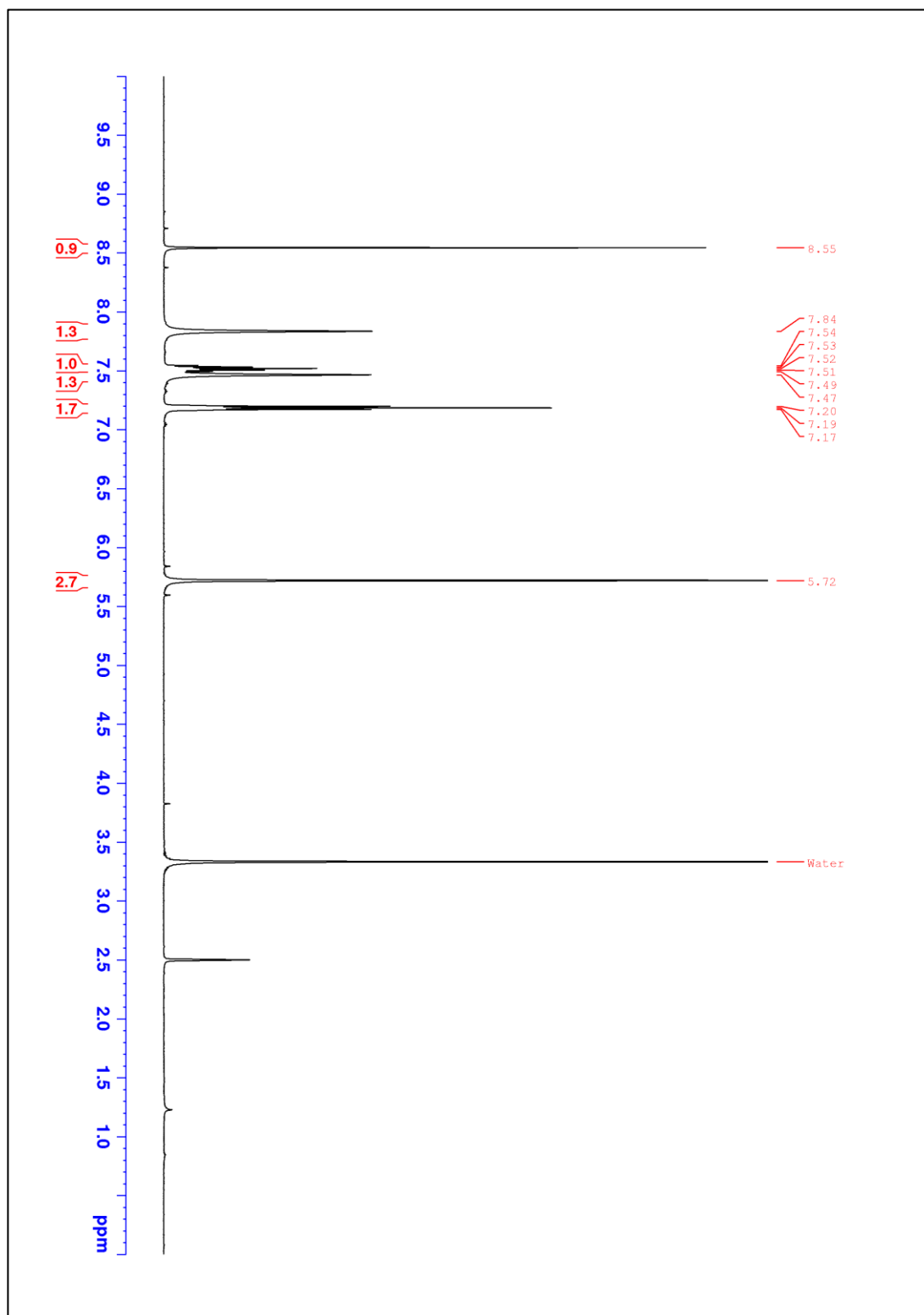


Figure 99: ¹H NMR of rufinamide 2 produced manually

APPENDIX

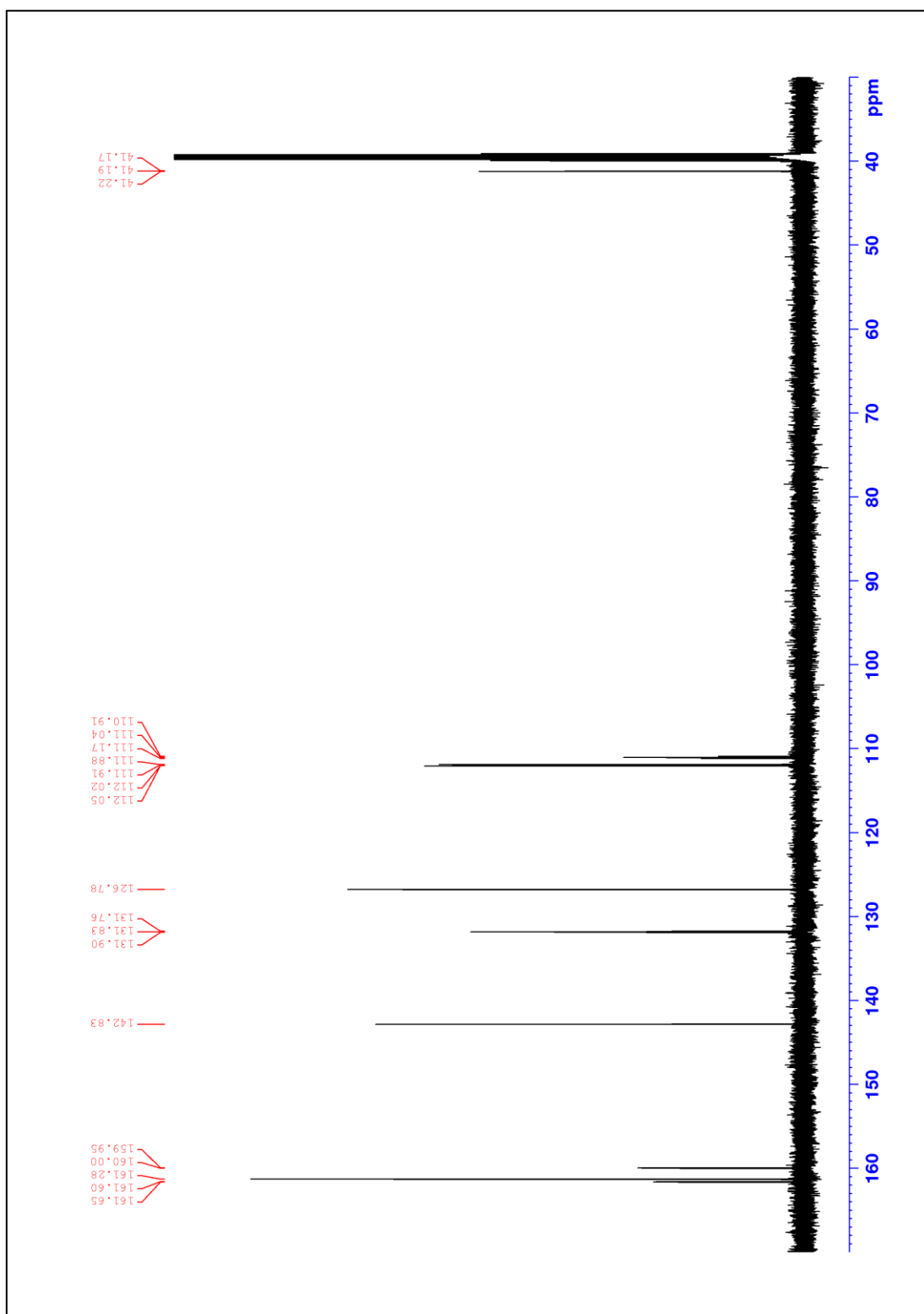


Figure 100: ^{13}C NMR (DEPTQ) of rufinamide **2** produced manually

APPENDIX

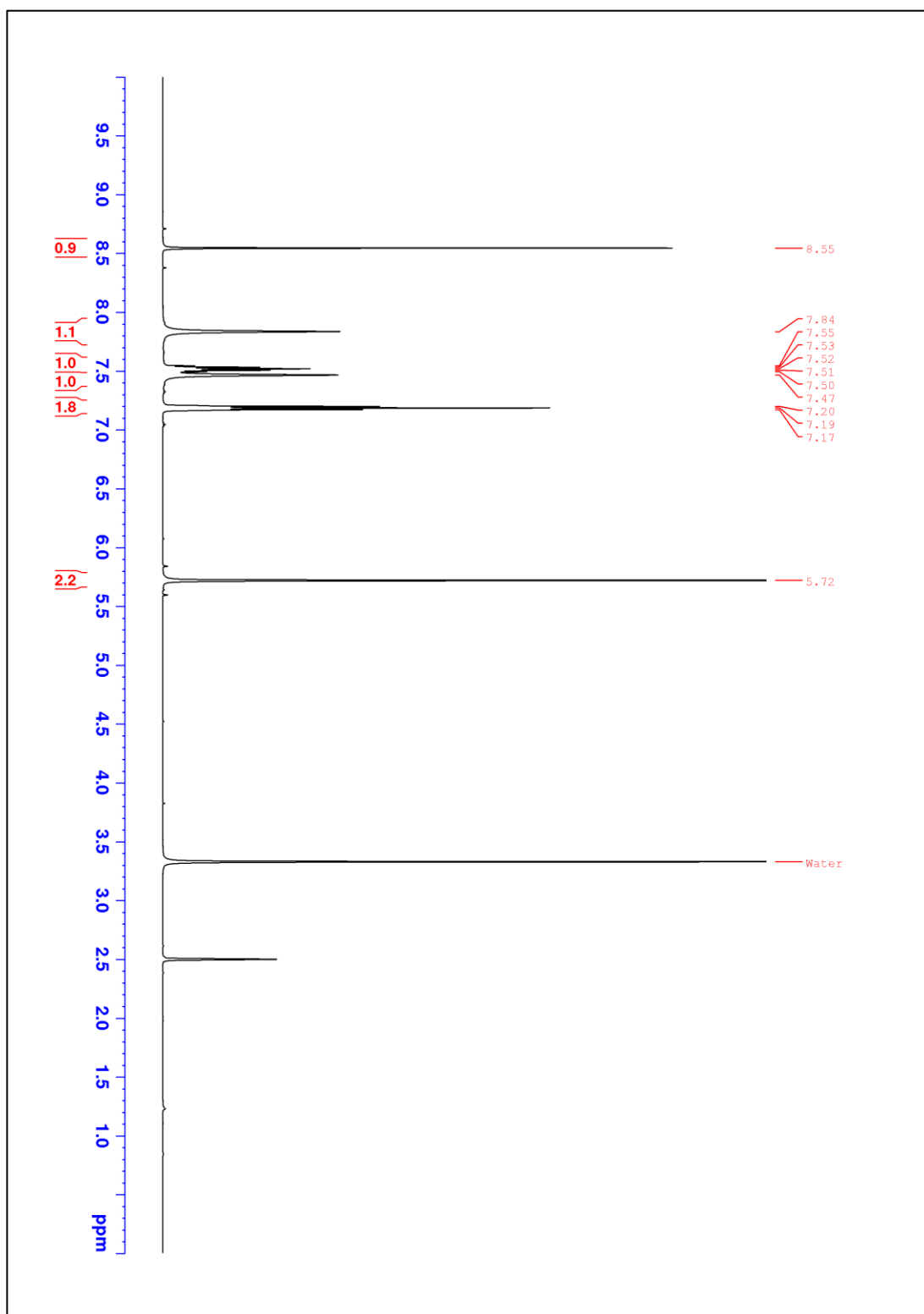


Figure 101: ^1H NMR of rufinamide 2 produced automatically

APPENDIX

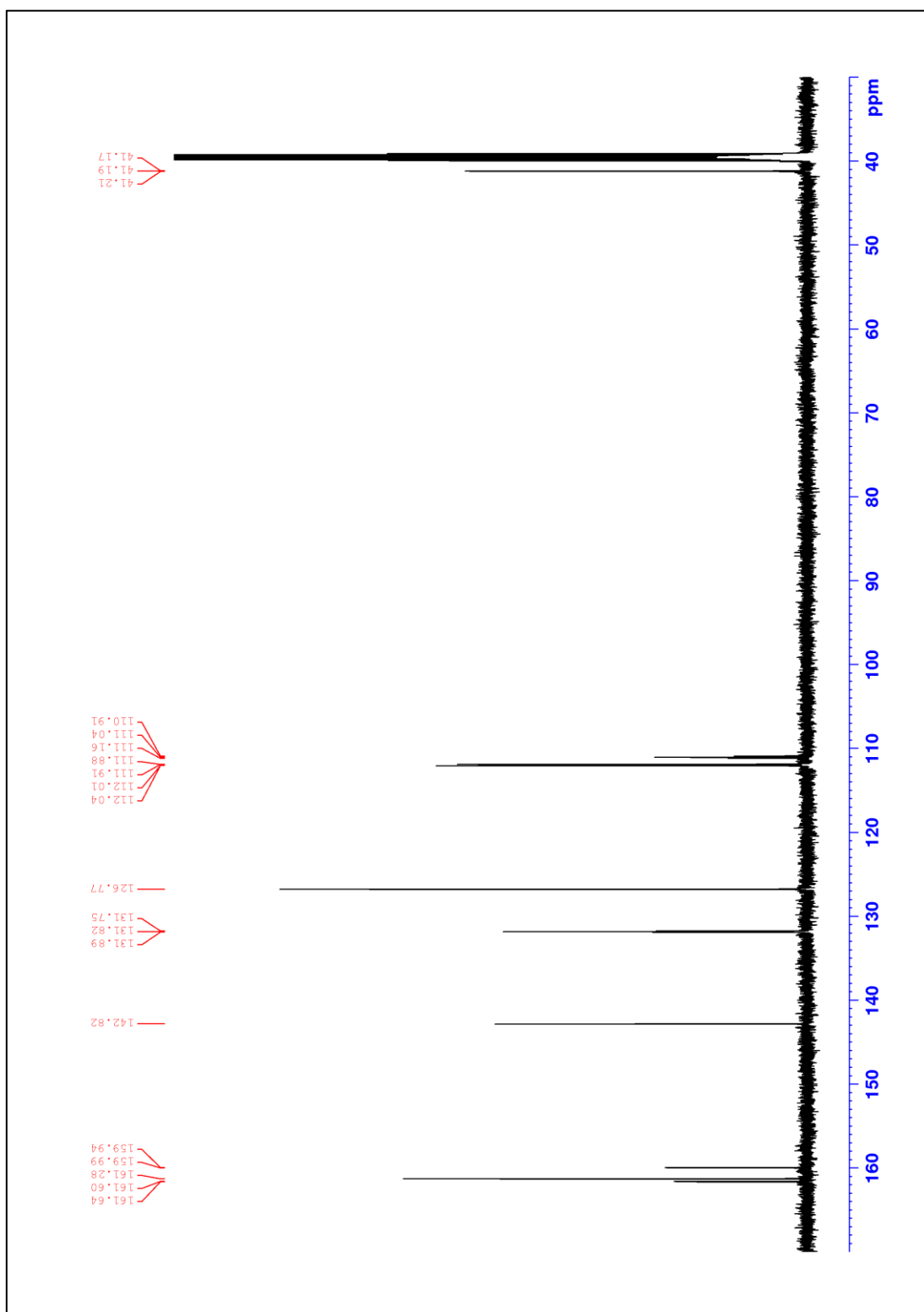


Figure 102: ^{13}C NMR (DEPTQ) of rufinamide **2** produced automatically

APPENDIX

I.III 1-[4-ETHOXY-3-(6,7-DIHYDRO-1-METHYL-7-OXO-3-PROPYL-1H-PYRAZOLO[4,3-D]PYRIMIDIN-5-YL)PHENYLSULFOYL]-4-METHYLPYPERAZINE
(SILDENAFIL) (3)

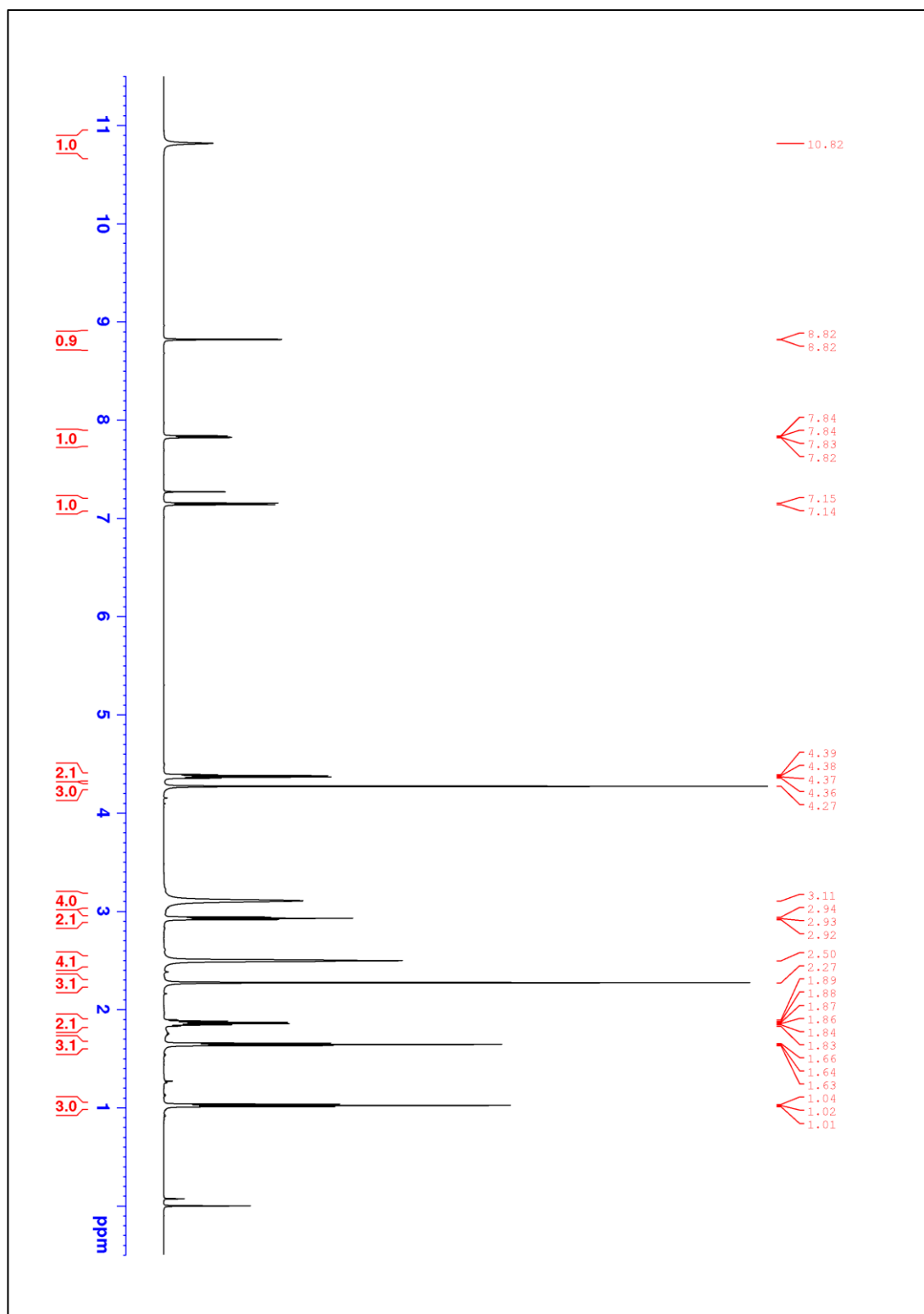


Figure 103: ¹H NMR of sildenafil 3 produced manually

APPENDIX

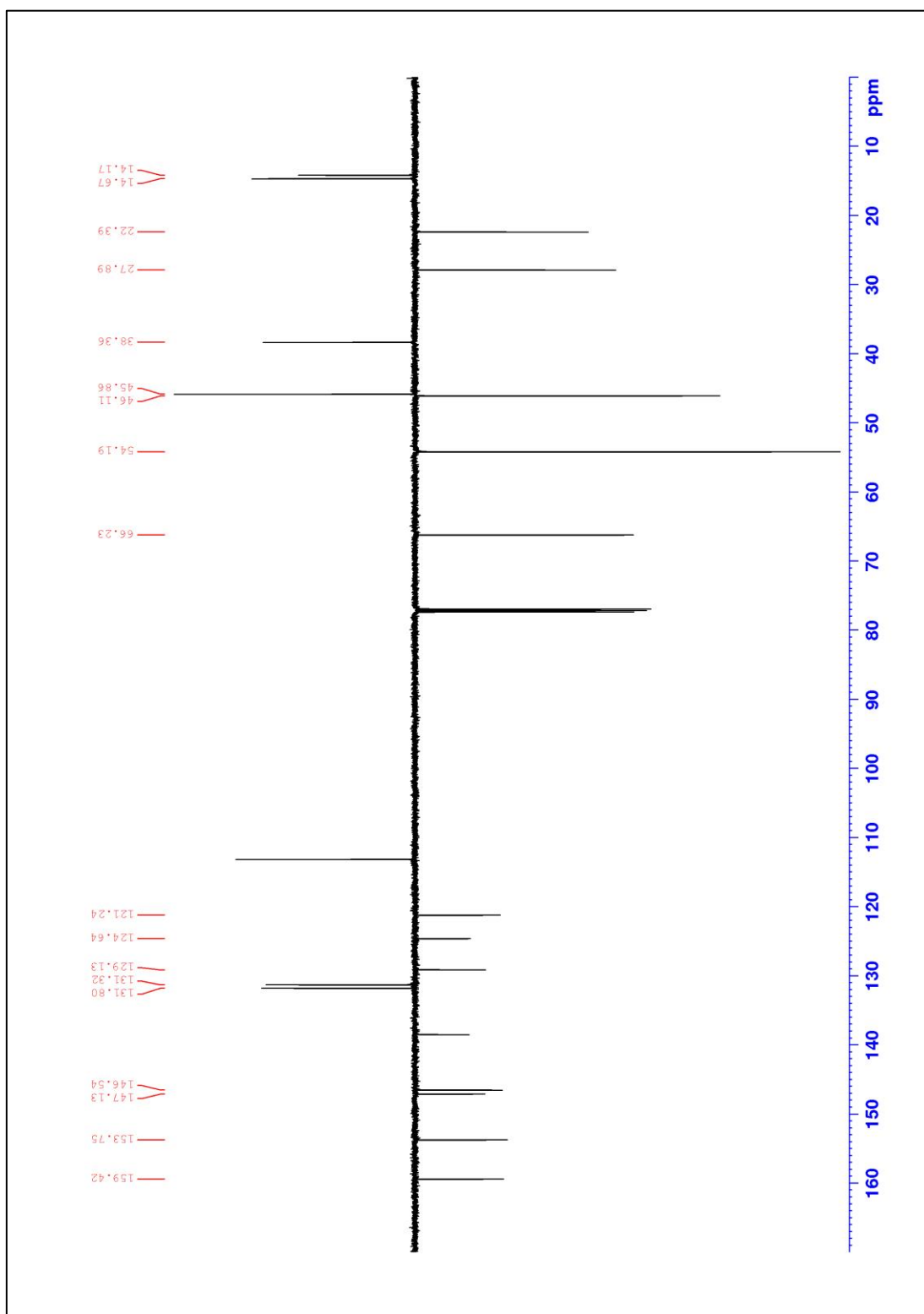


Figure 104: ¹³C NMR (DEPTQ) of sildenafil **3** produced manually

APPENDIX

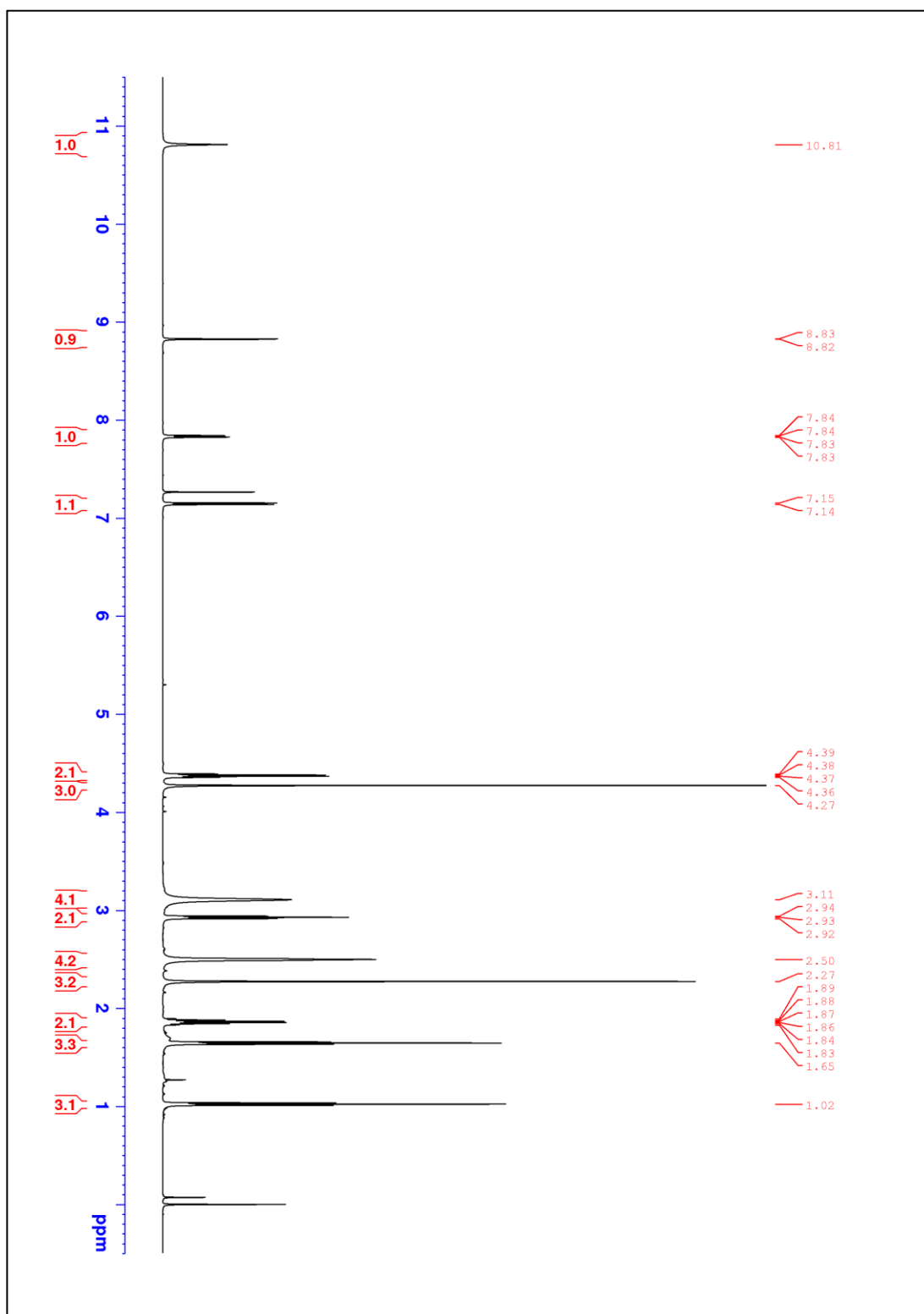


Figure 105: ¹H NMR of sildenafil 3 produced automatically

APPENDIX

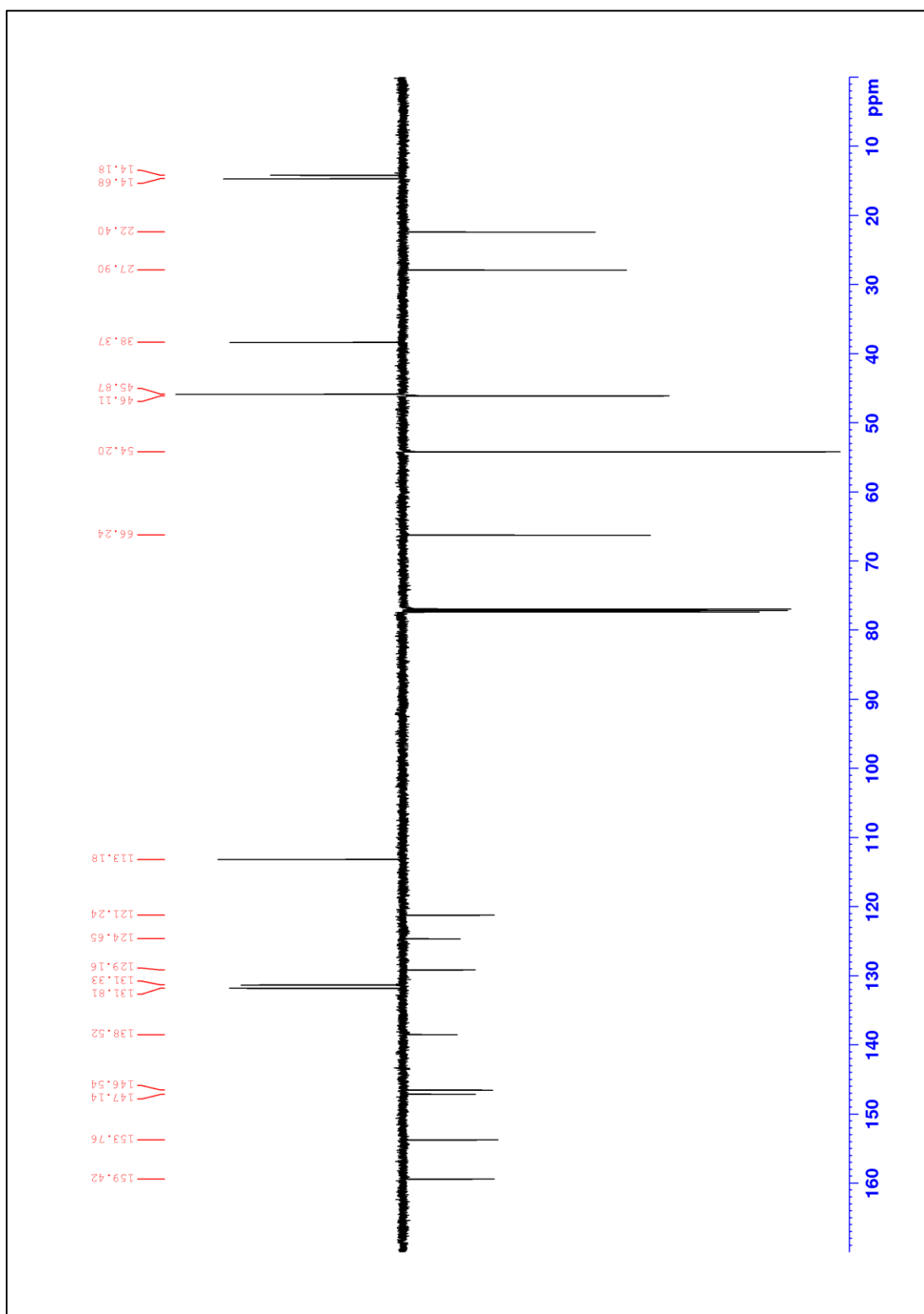
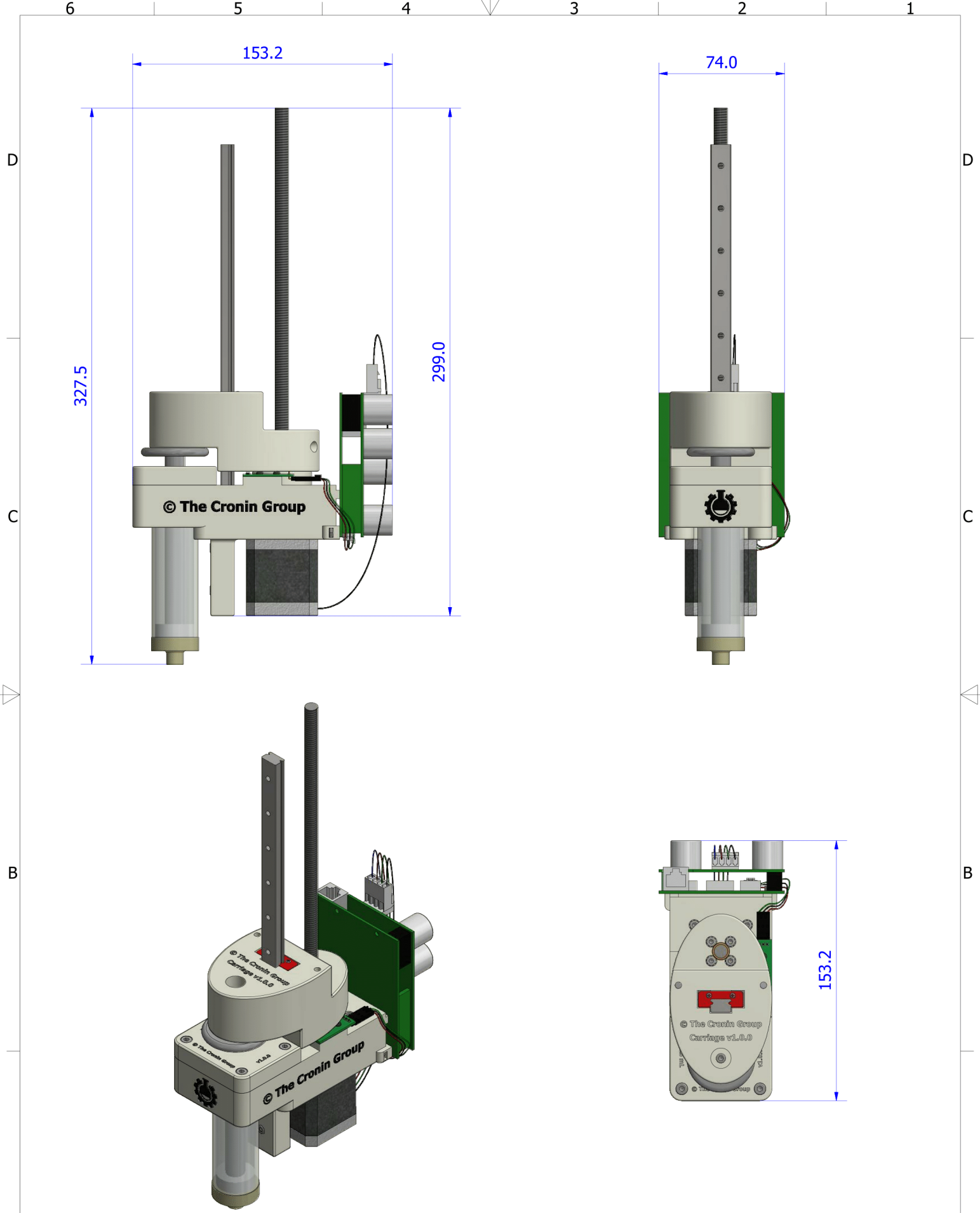


Figure 106: ^{13}C NMR (DEPTQ) of sildenafil **3** produced automatically

APPENDIX

II ENGINEERING DRAWINGS

APPENDIX



ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ± 0.25 mm. CENTRE DISTANCE ± 0.25 mm. ANGLES $0^{\circ}30''$.	
TOLERANCE CLASS:	TO SUPPLIER SPEC
MATERIAL:	
SPEC:	
FINISH:	
WEIGHT:	N/A

DRAWN BY:
Sebastian Steiner



© CRONIN GROUP
UNIVERSITY OF GLASGOW

REVIEWED BY: Graham Sharp
APPROVED BY: Lee Cronin

THIS WORK IS LICENCED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

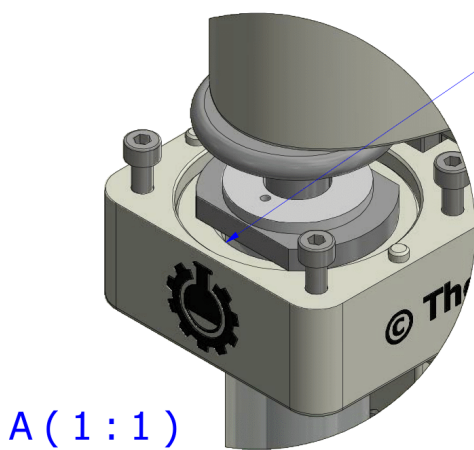
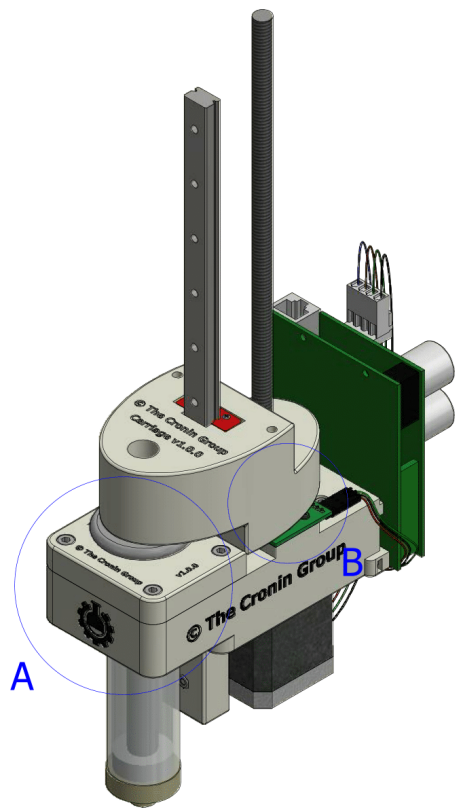
DATE:	17/11/2017	SHEET	1/4	SIZE	A3	SCALE	1 : 2
-------	------------	-------	-----	------	----	-------	-------

UNCONTROLLED COPY IF PRINTED
DO NOT SCALE: IF IN DOUBT, ASK

DESCRIPTION:
Chemputer Pump Assembly w/ 25mL Syringe

PART NO:	Pump Assembly	VERSION:	1.0.0
----------	---------------	----------	-------

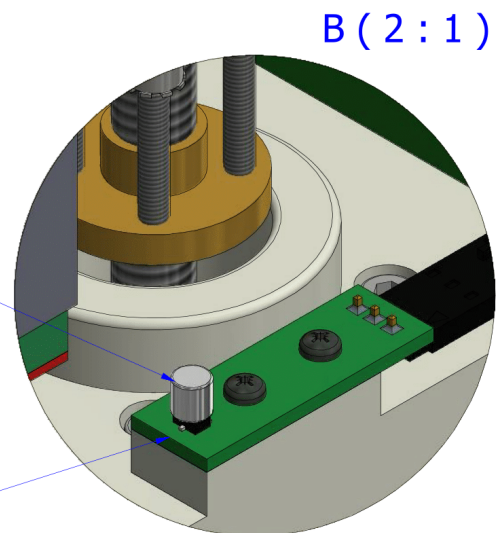
PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	Pump Motor Connector	Base element for the pump
2	1	Pump Carriage	Carriage for Chemputer pump
3	1	25mL Plate	Top plate for 25mL syringes
4	1	25mL Sleeve	Sleeve insert for the 25mL syringe
5	1	25mL syringe	
6	1	Carriage	MGN 12 square block carriage
7	1	Guide rail	MGNR12 guide rail
8	1	17HSL5415-252-1	NEMA17 motor with lead screw
9	1	Lead screw nut	Tr8*4 (P2) nut with four M3 mounting holes
10	1	Hall Sensor Breakout	Breakout board for Hall effect sensor
11	2	Magnet	4mm O.D. neodymium magnet
12	1	PCB	Old PoE / stepper control PCB
13	2	ISO 7045 - M2 x 10 - 4.8 - Z	Cross Recessed Pan Head Machine Screw
14	2	DIN 912 - M3 x 8	Cylinder Head Cap Screw
15	7	DIN 912 - M3 x 10	Cylinder Head Cap Screw
16	4	DIN 912 - M3 x 20	Cylinder Head Cap Screw
17	4	DIN 912 - M3 x 30	Cylinder Head Cap Screw
18	4	DIN 912 - M4 x 30	Cylinder Head Cap Screw
19	5	DIN 934 - M3	Hex Nut
20	15	DIN 6798 - A 3.2	Serrated Lock Washer
21	4	M4 Kalei Nut	Press-in hex nut
22	1	ANSI B18.3 - No. 6 - 32 UNC - 3/4 HS HCS	Hexagon Socket Head Cap Screw
23	1	TE Connectivity SL-156	3.962mm Pitch, 4 Way, 1 Row Female Straight Connector
24	1	Molex MICROCLASP 51382	2mm Pitch, 4 Way, 1 Row Female Straight Connector
25	1	Molex SL 70066	3 Way SIL connector



Part 4: Sleeve

Part 11: Magnet

Part 10: Hall Sensor Breakout



B (2:1)

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm. CENTRE DISTANCE ±0.25mm. ANGLES 0°30".	
TOLERANCE CLASS:	TO SUPPLIER SPEC
MATERIAL:	
SPEC:	
FINISH:	
WEIGHT:	N/A

DRAWN BY:
Sebastian Steiner



CRONIN GROUP
COMPLEX CHEMICAL SYSTEMS

© CRONIN GROUP
UNIVERSITY OF GLASGOW

REVIEWED BY: Graham Sharp
APPROVED BY: Lee Cronin

THIS WORK IS LICENCED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

DATE:
17/11/2017

SHEET
2/4

SIZE
A3

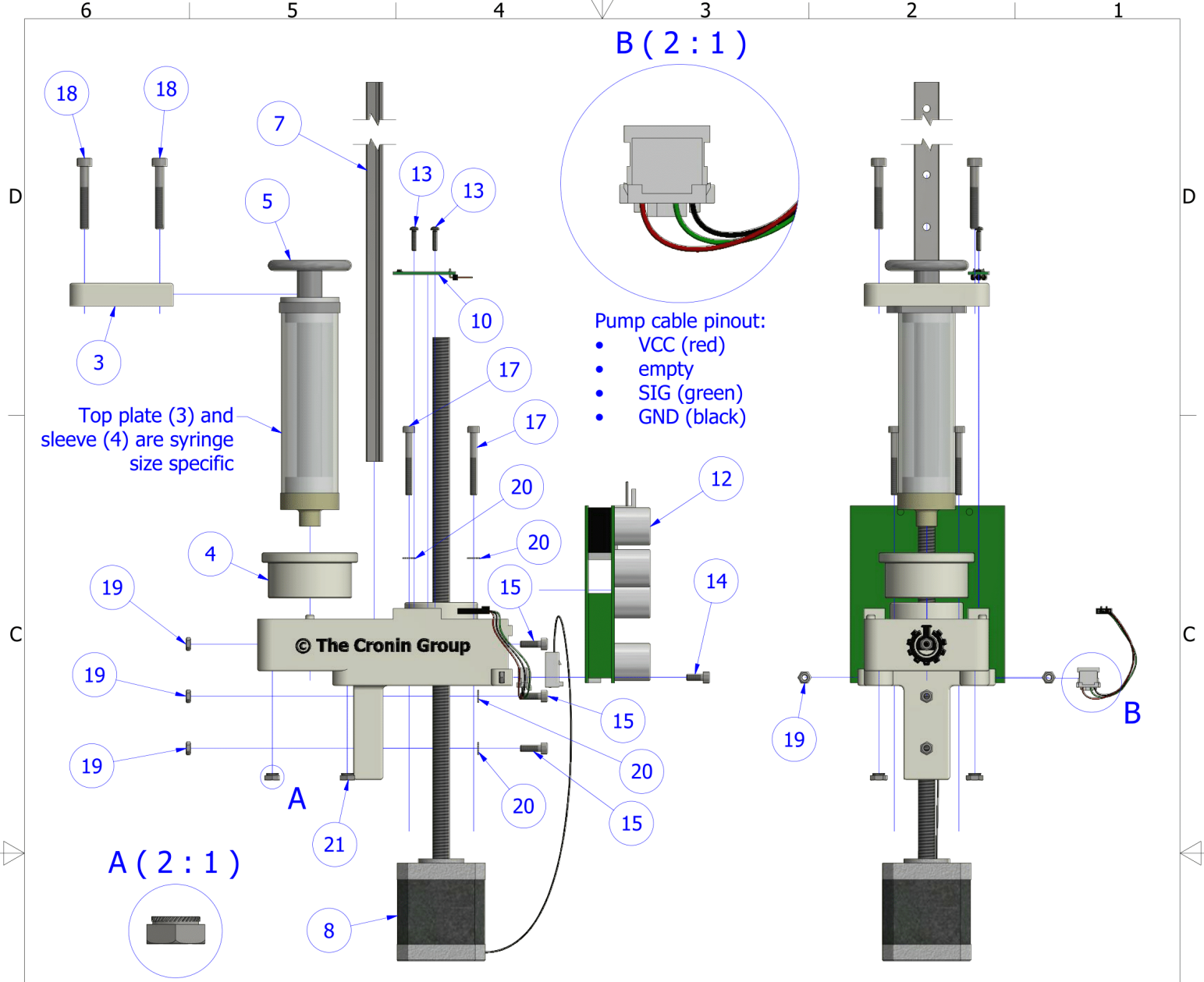
SCALE
1 : 2

UNCONTROLLED COPY IF PRINTED
DO NOT SCALE: IF IN DOUBT, ASK

DESCRIPTION:
Chemputer Pump Assembly
w/ 25mL Syringe

PART NO:
Pump Assembly

VERSION:
1.0.0

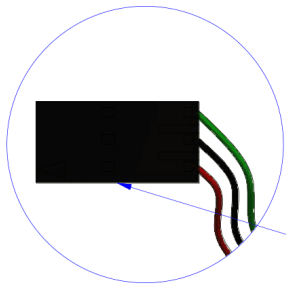


Top plate (3) and sleeve (4) are syringe size specific

- Pump cable pinout:
- VCC (red)
 - empty
 - SIG (green)
 - GND (black)

Kalei Nut (22) presses into the body and stays firmly seated

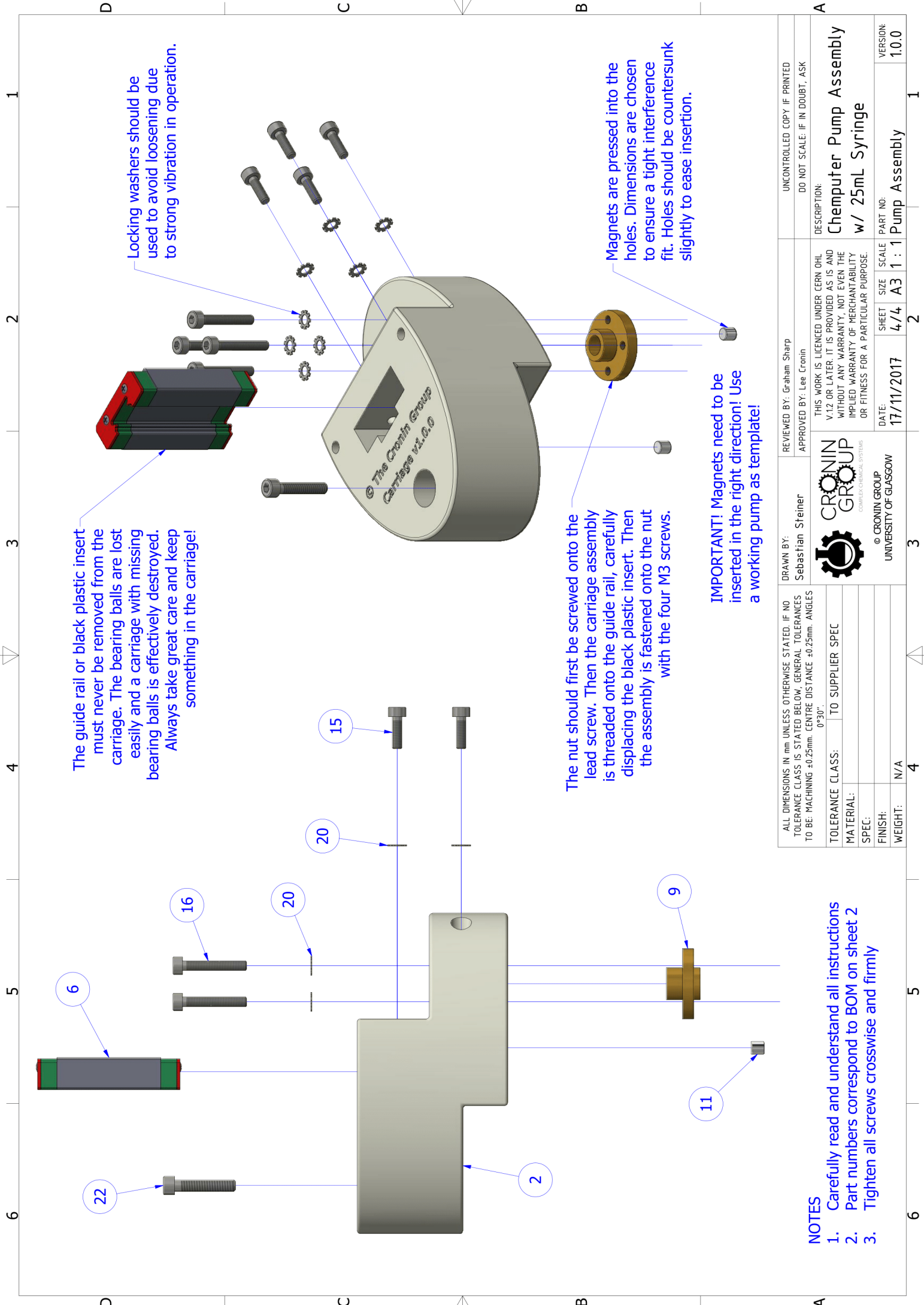
M2x10 are directly screwed into the body without tapping. Make sure the holes are properly cleaned before!



Ensure correct pinout as shown here! VCC, GND and SIG are marked on the sensor breakout board. Do not flip the connector!

Isometric view (1 : 4)

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm. CENTRE DISTANCE ±0.25mm. ANGLES 0°30".		DRAWN BY: Sebastian Steiner		REVIEWED BY: Graham Sharp		UNCONTROLLED COPY IF PRINTED	
TOLERANCE CLASS: TO SUPPLIER SPEC		© CRONIN GROUP UNIVERSITY OF GLASGOW		APPROVED BY: Lee Cronin		DO NOT SCALE: IF IN DOUBT, ASK	
MATERIAL:				THIS WORK IS LICENCED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.		DESCRIPTION: Chemputer Pump Assembly w/ 25mL Syringe	
SPEC:							
FINISH:							
WEIGHT:	N/A						
		DATE:	17/11/2017	SHEET:	3/4	SIZE:	A3
		SCALE:	1 : 2	PART NO:	Pump Assembly	VERSION:	1.0.0



The guide rail or black plastic insert must never be removed from the carriage. The bearing balls are lost easily and a carriage with missing bearing balls is effectively destroyed. Always take great care and keep something in the carriage!

Locking washers should be used to avoid loosening due to strong vibration in operation.


The nut should first be screwed onto the lead screw. Then the carriage assembly is threaded onto the guide rail, carefully displacing the black plastic insert. Then the assembly is fastened onto the nut with the four M3 screws.

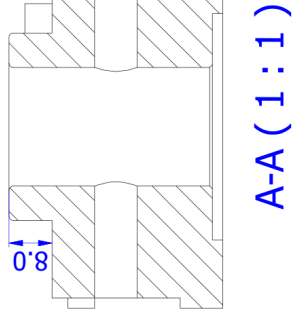
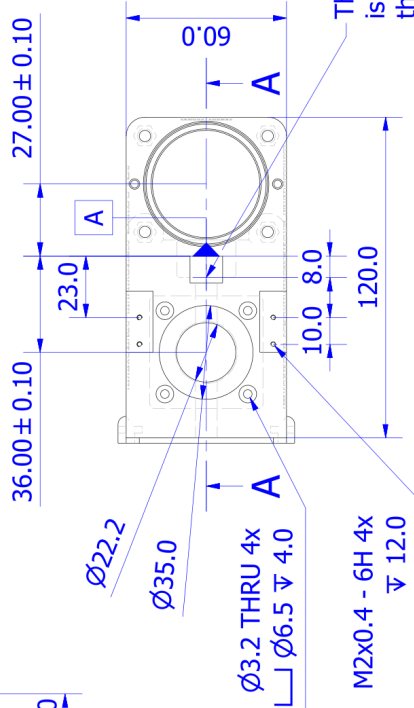
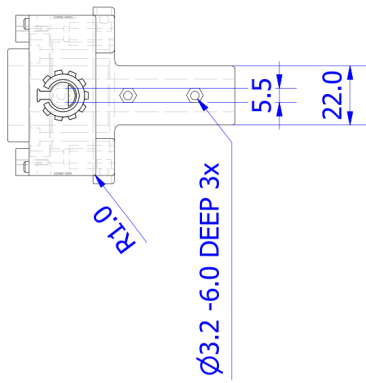
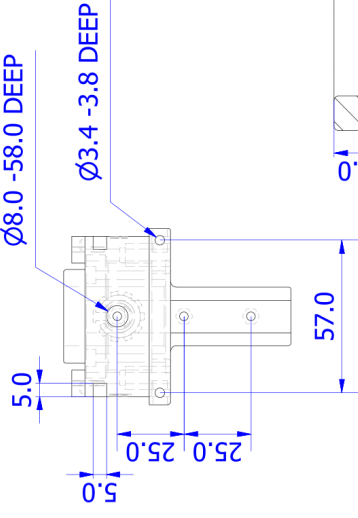
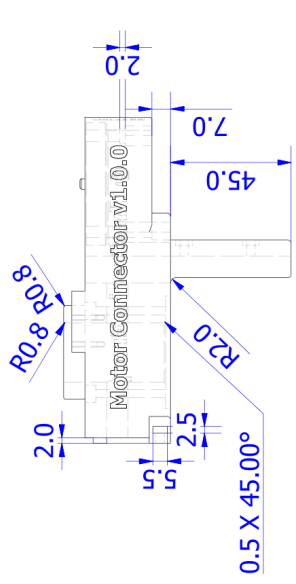
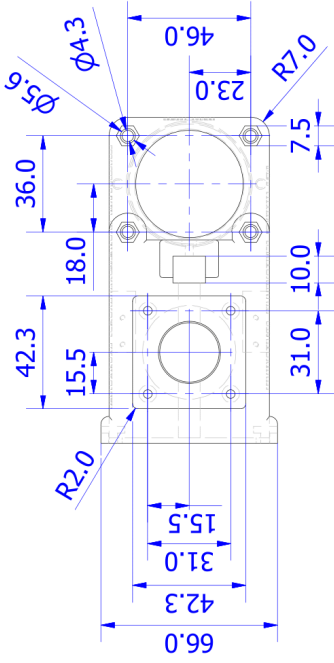
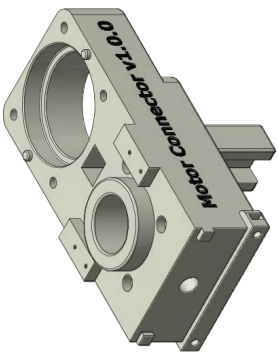
Magnets are pressed into the holes. Dimensions are chosen to ensure a tight interference fit. Holes should be countersunk slightly to ease insertion.

IMPORTANT! Magnets need to be inserted in the right direction! Use a working pump as template!

NOTES

1. Carefully read and understand all instructions
2. Part numbers correspond to BOM on sheet 2
3. Tighten all screws crosswise and firmly

UNCONTROLLED COPY IF PRINTED DO NOT SCALE: IF IN DOUBT, ASK	
REVIEWED BY: Graham Sharp APPROVED BY: Lee Cronin	DESCRIPTION: Chemputer Pump Assembly w/ 25mL Syringe
THIS WORK IS LICENSED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.	PART NO: 1 Pump Assembly
CRONIN GROUP CONSTRUCTIONAL SYSTEMS	SHEET SCALE 4/4 A3 1 : 1
DATE: 17/11/2017	VERSION: 1.0.0
DRAWN BY: Sebastian Steiner	
 © CRONIN GROUP UNIVERSITY OF GLASGOW	
ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°-30°.	
TOLERANCE CLASS: TO SUPPLIER SPEC	
MATERIAL:	
SPEC:	
FINISH:	
WEIGHT: N/A	



The center point of the base rectangle is coincident with the center point of the carriage. The guide rail mount is dimensioned in a way that the guide rail face lies on the XZ plane.

NOTES

1. Outer edges should be broken by fillets or chamfers
2. Part number, revision and copyright must be embossed on the sides
3. A 3D model is available

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°30'.

TOLERANCE CLASS:	TO SUPPLIER SPEC
MATERIAL:	RDG720
SPEC:	Smooth - Ivory
FINISH:	GLOSSY
WEIGHT:	200.9 g

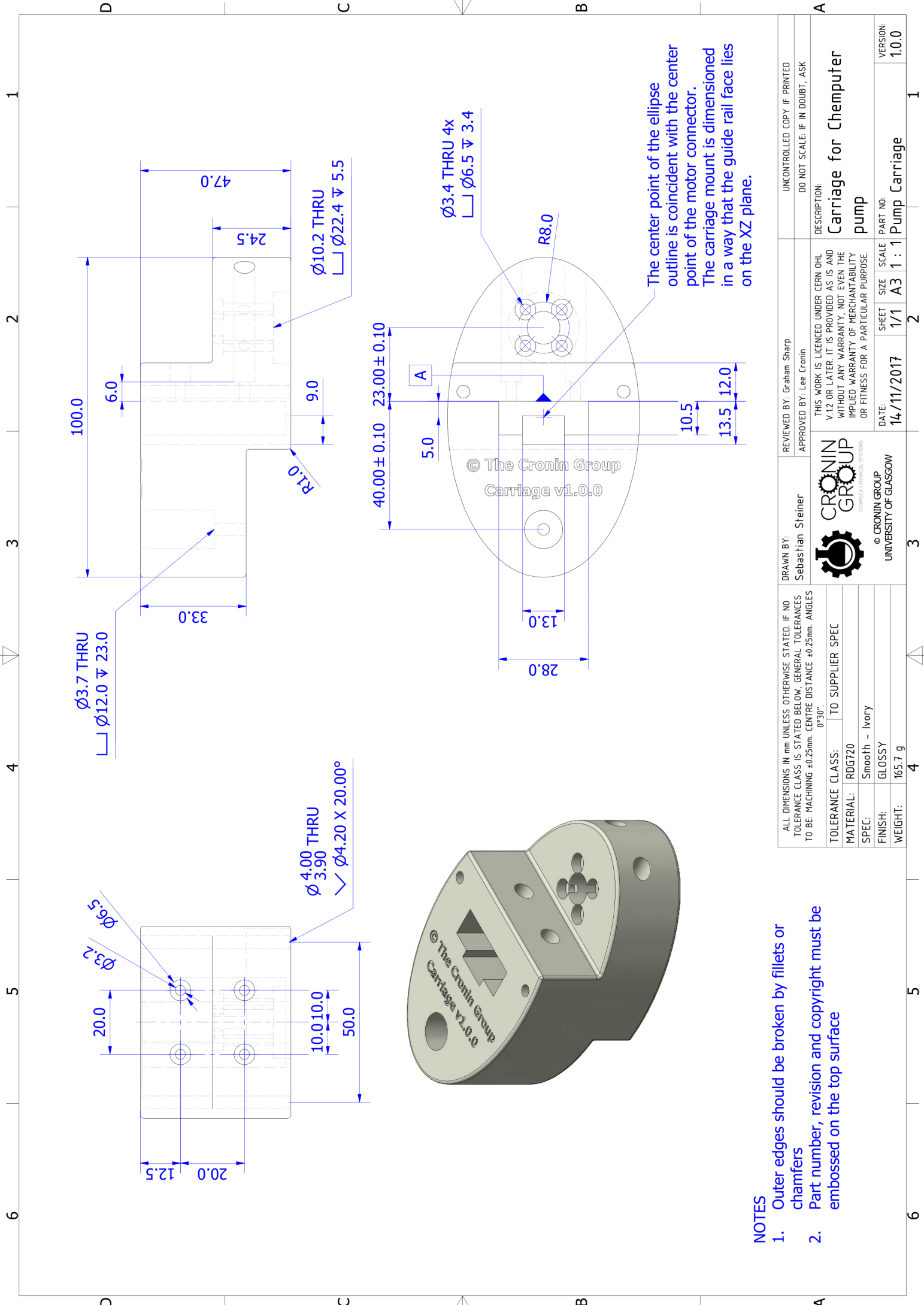
DRAWN BY: Sebastian Steiner
 REVIEWED BY: Graham Sharp
 APPROVED BY: Lee Cronin



DATE: 14/11/2017
 SHEET: 1/1
 SCALE: A3
 SIZE: 1 : 2
 PART NO.: Pump Motor Connector
 VERSION: 1.0.0

UNCONTROLLED COPY IF PRINTED
 DO NOT SCALE: IF IN DOUBT, ASK

DESCRIPTION:
Base element for the pump

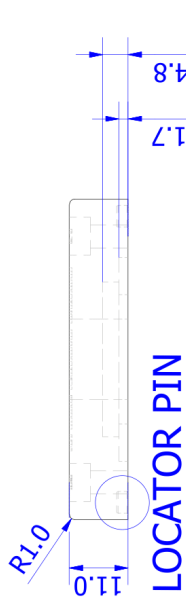


NOTES

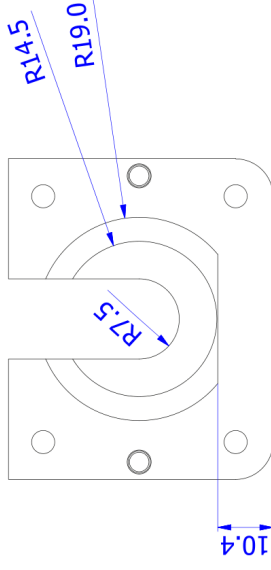
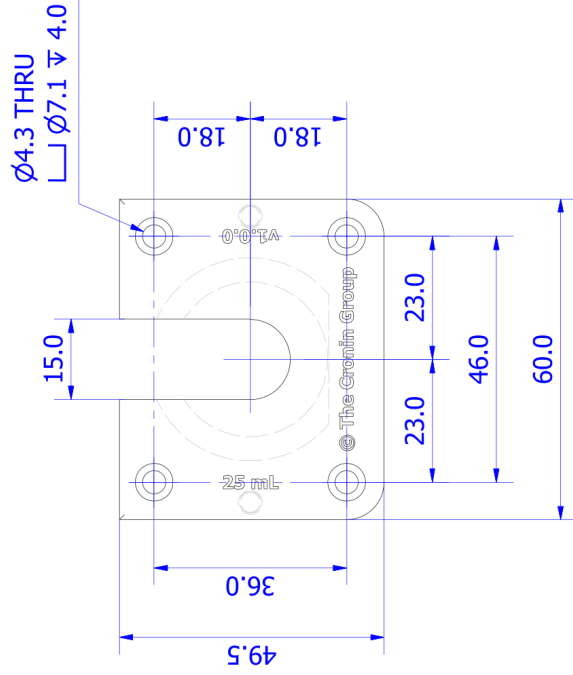
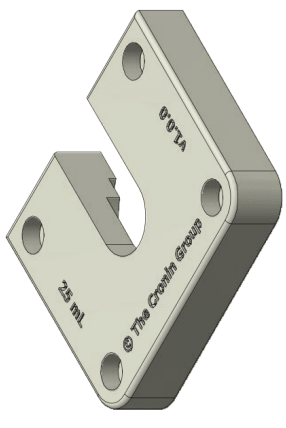
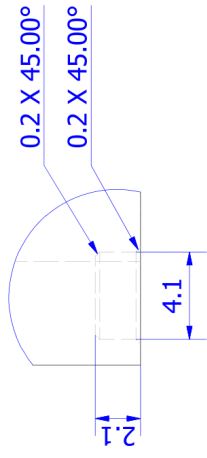
- Outer edges should be broken by fillets or chamfers
- Part number, revision and copyright must be embossed on the top surface

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°-30°.	DRAWN BY: Sebastian Steiner	REVIEWED BY: Graham Sharp APPROVED BY: Lee Cronin	UNCONTROLLED COPY IF PRINTED DO NOT SCALE: IF IN DOUBT, ASK
	TOLERANCE CLASS: TO SUPPLIER SPEC	MATERIAL: RDG720	FINISH: Glossy
SPEC: Smooth - Ivory	WEIGHT: 165.7 g	DATE: 14/11/2017	PART NO: 1 Pump Carriage
FINISH: GLOSSY	WEIGHT: 165.7 g	SHEET: 1/1	SCALE: A3 1 : 1
FINISH: GLOSSY	WEIGHT: 165.7 g	SHEET: 2	VERSION: 1.0.0





**LOCATOR PIN (4 : 1)
TYPICAL 2 PLACES**

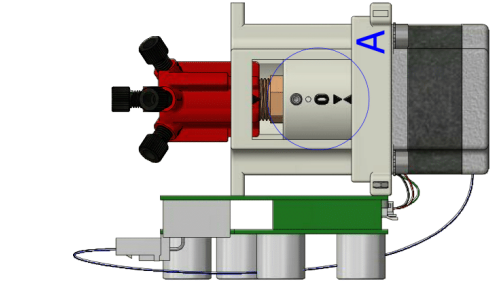
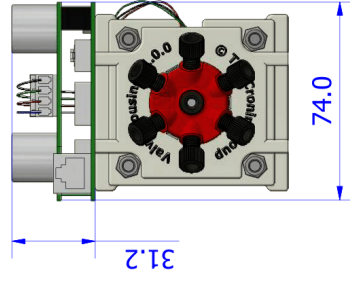
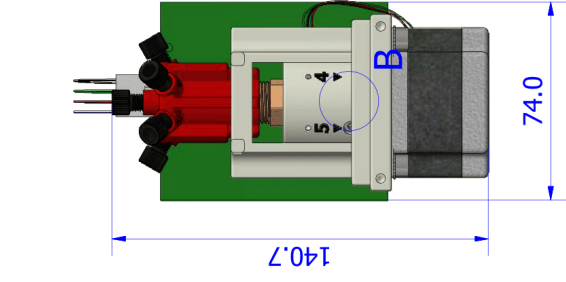


NOTES

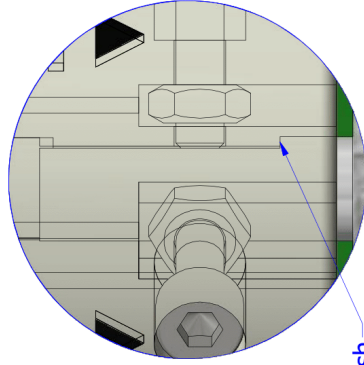
1. Outer edges should be broken by fillets or chamfers
2. Syringe volume, revision and copyright must be embossed on the top surface

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°-30°.	DRAWN BY: Sebastian Steiner	REVIEWED BY: Graham Sharp APPROVED BY: Lee Cronin	UNCONTROLLED COPY IF PRINTED DO NOT SCALE: IF IN DOUBT, ASK
	TOLERANCE CLASS: TO SUPPLIER SPEC MATERIAL: RDG720 SPEC: Smooth - Ivory FINISH: GLOSSY WEIGHT: 28.5 g		DESCRIPTION: Top plate for 25mL syringes
© CRONIN GROUP UNIVERSITY OF GLASGOW		DATE: 14/11/2017	PART NO: 25mL Plate
SHEET: 1/1 SCALE: A3		SHEET: 2	VERSION: 1.0.0

PARTS LIST			
ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	Valve Housing	Housing for Chemputer valve
2	1	Valve Motor Connector	Motor connector for Chemputer Valve
3	1	V-340	Six-way selection valve
4	1	23HS0420	NEMA23 motor with D cut shaft
5	1	Hall Sensor Breakout	Breakout board for Hall effect sensor
6	6	Magnet	2mm O.D. neodymium magnet
7	1	PCB	Old PoE / stepper control PCB
8	2	DIN 912 - M3 x 8	Cylinder Head Cap Screw
9	3	DIN 912 - M3 x 12	Cylinder Head Cap Screw
10	1	DIN 914 - M4 x 8	Hexagon Socket Set Screw
11	4	DIN 912 - M5 x 16	Cylinder Head Cap Screw
12	5	DIN 934 - M3	Hex Nut
13	1	DIN 562 - M4	Square Nut
14	4	DIN 934 - M5	Hex Nut
15	4	DIN 6798 - A 5.3	Serrated Lock Washer
16	1	TE Connectivity SL-156	3.962mm Pitch, 4 Way, 1 Row Female Straight Connector
17	1	Molex MICROCLASP 51382	2mm Pitch, 4 Way, 1 Row Female Straight Connector
18	1	Molex SL 70066	3 Way SIL connector

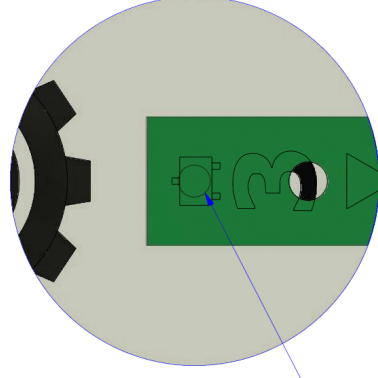


B (3 : 1)



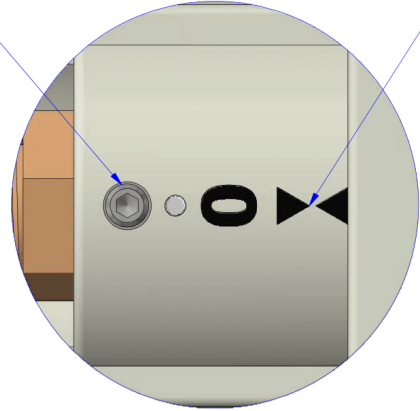
Motor connector sits flush on D cut to ensure vertical alignment of the assembly

C (3 : 1)



SMD Hall effect sensor and positional magnet line up vertically

A (2 : 1)



Set screw holding valve shaft

Alignment indicator

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°-30°.

TOLERANCE CLASS:	TO SUPPLIER SPEC
MATERIAL:	
SPEC:	
FINISH:	
WEIGHT:	N/A

DRAWN BY:
Sebastian Steiner



© CRONIN GROUP
UNIVERSITY OF GLASGOW

REVIEWED BY: Graham Sharp
APPROVED BY: Lee Cronin

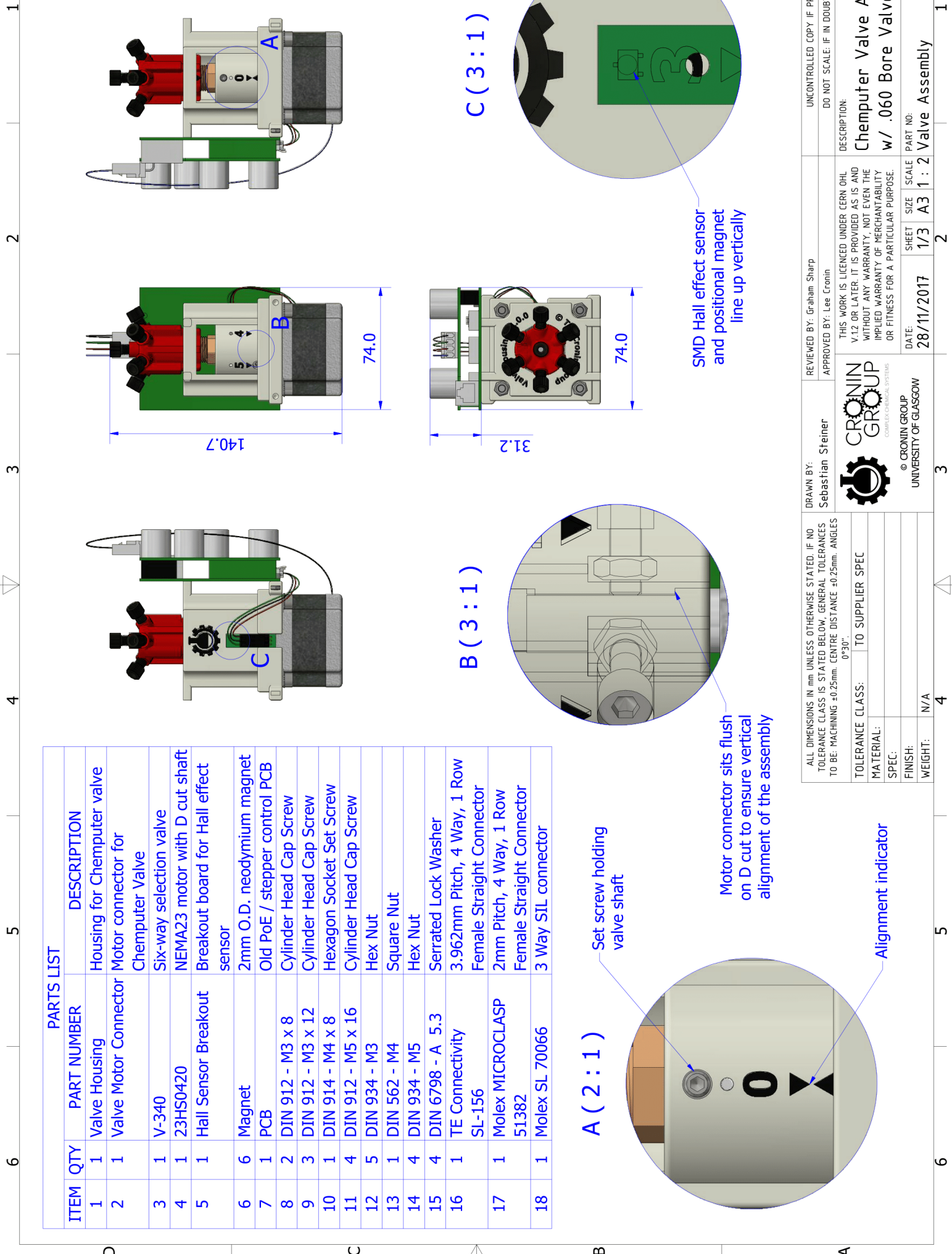
THIS WORK IS LICENSED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

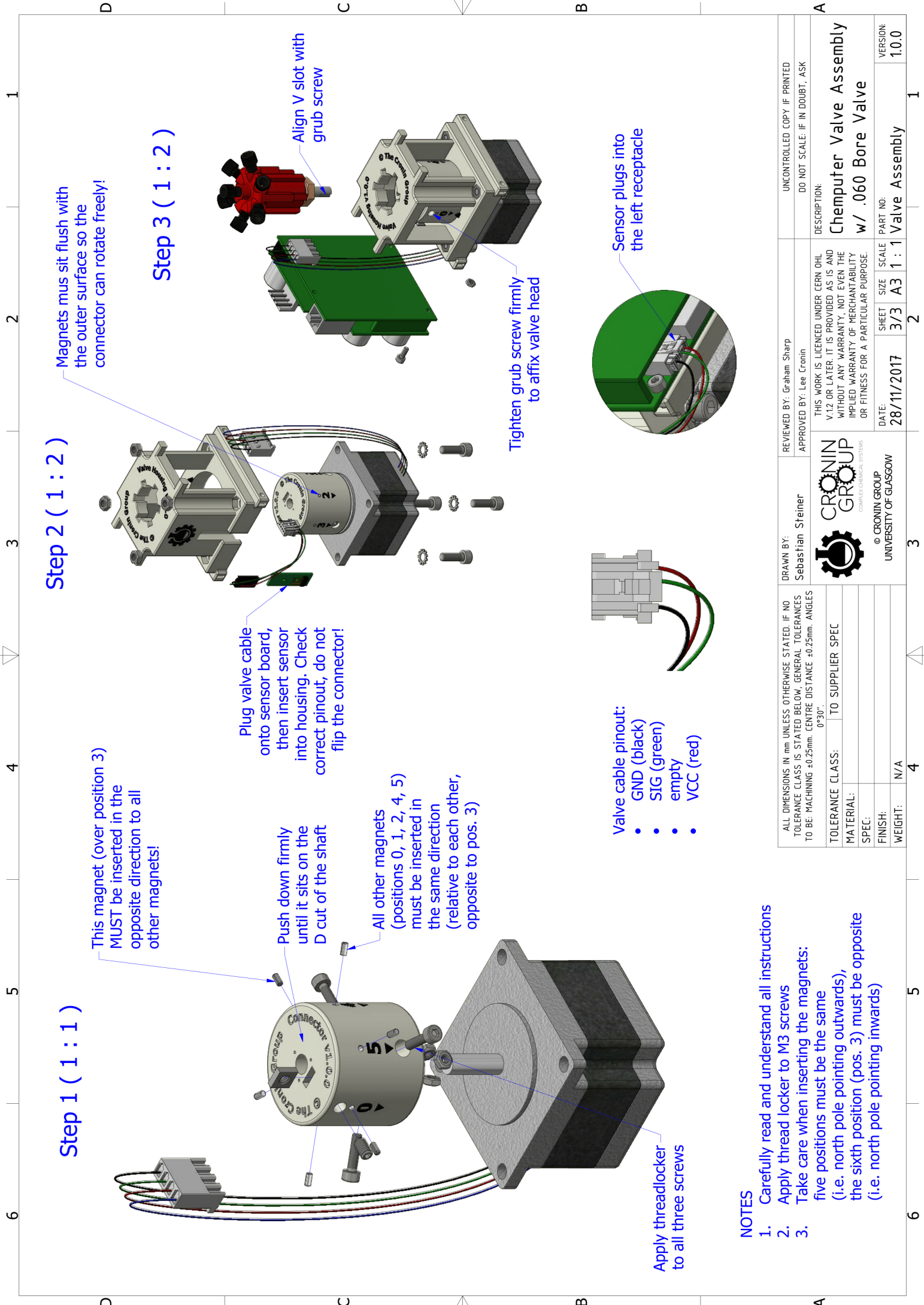
DATE: 28/11/2017
SHEET: 1/3
SCALE: A3
SIZE: A3
PART NO: 1 : 2 Valve Assembly

UNCONTROLLED COPY IF PRINTED
DO NOT SCALE: IF IN DOUBT, ASK

DESCRIPTION:
**Chemputer Valve Assembly
w/ .060 Bore Valve**

VERSION:
1.0.0





Step 1 (1 : 1)

This magnet (over position 3) MUST be inserted in the opposite direction to all other magnets!

Push down firmly until it sits on the D cut of the shaft

All other magnets (positions 0, 1, 2, 4, 5) must be inserted in the same direction (relative to each other, opposite to pos. 3)

Apply threadlocker to all three screws

Step 2 (1 : 2)

Magnets must sit flush with the outer surface so the connector can rotate freely!

Plug valve cable onto sensor board, then insert sensor into housing. Check correct pinout, do not flip the connector!

Tighten grub screw firmly to affix valve head

Step 3 (1 : 2)

Align V slot with grub screw

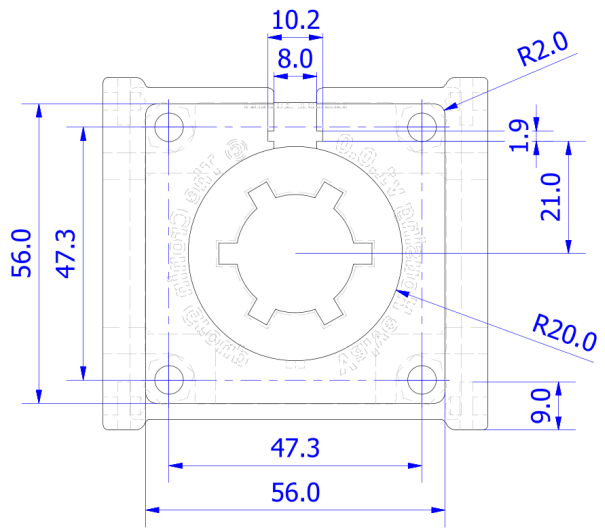
Sensor plugs into the left receptacle

- Valve cable pinout:
- GND (black)
 - SIG (green)
 - empty
 - VCC (red)

NOTES

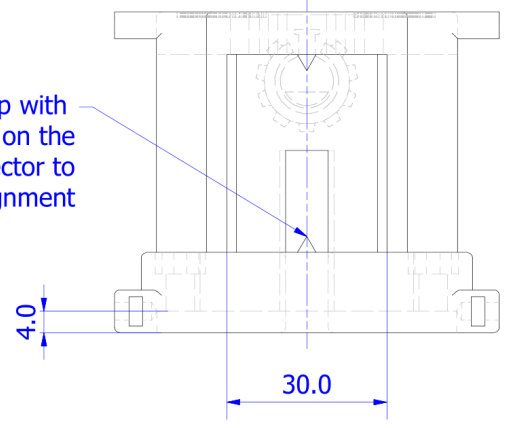
- Carefully read and understand all instructions
- Apply thread locker to M3 screws
- Take care when inserting the magnets: five positions must be the same (i.e. north pole pointing outwards), the sixth position (pos. 3) must be opposite (i.e. north pole pointing inwards)

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°-30°.		DRAWN BY: Sebastian Steiner	REVIEWED BY: Graham Sharp APPROVED BY: Lee Cronin	UNCONTROLLED COPY IF PRINTED DO NOT SCALE: IF IN DOUBT, ASK
TOLERANCE CLASS: TO SUPPLIER SPEC	MATERIAL:	DESCRIPTION: Chemputer Valve Assembly w/ .060 Bore Valve		
SPEC:	FINISH: N/A	DATE: 28/11/2017	SHEET: 3/3	SCALE: A3
WEIGHT:	N/A	PART NO: 1 : 1	Valve Assembly	VERSION: 1.0.0



A (1 : 1)

Arrow lines up with counterpart on the motor connector to indicate alignment

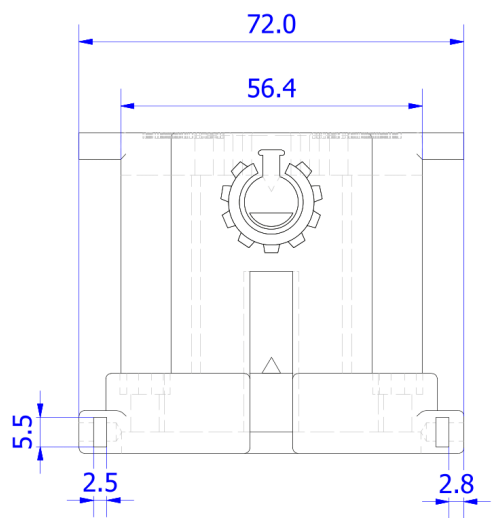


D

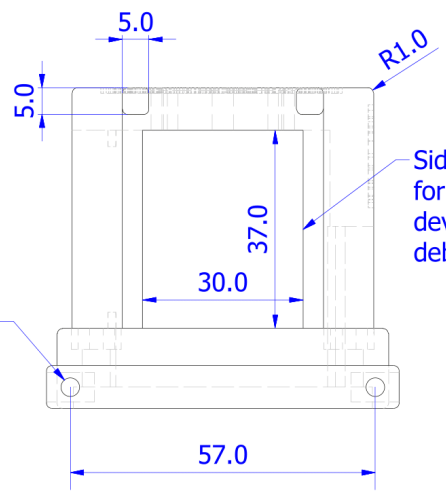
D

C

C



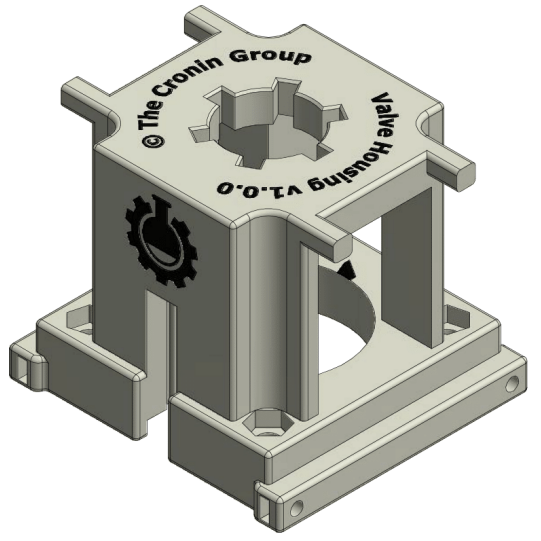
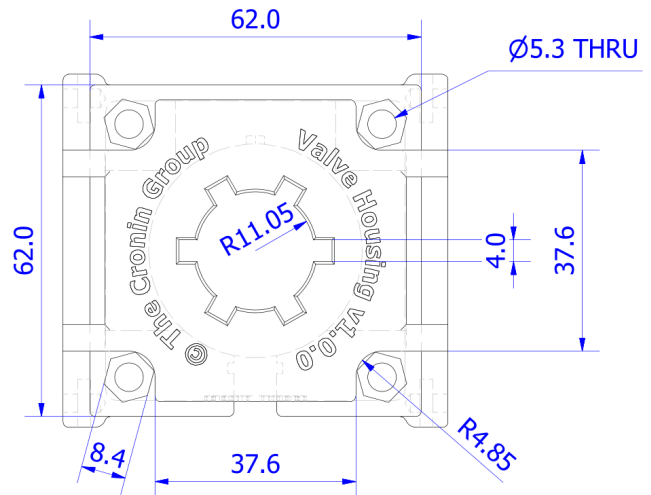
A
Ø3.4 -7.0 DEEP



Side and back windows for access during development and debugging

B

B



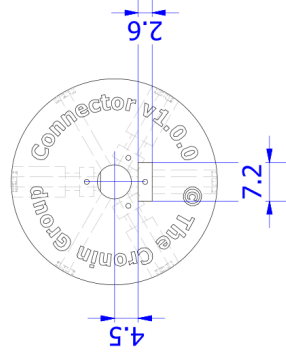
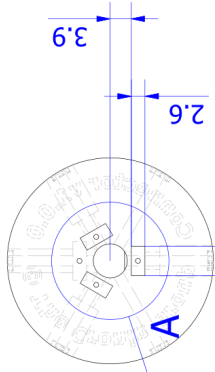
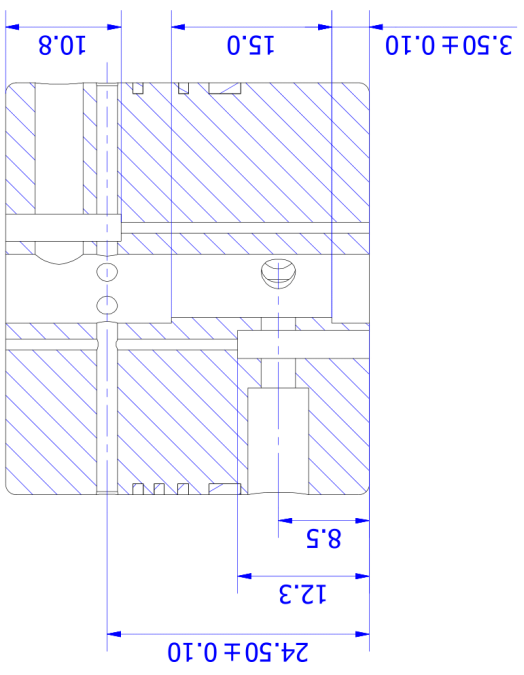
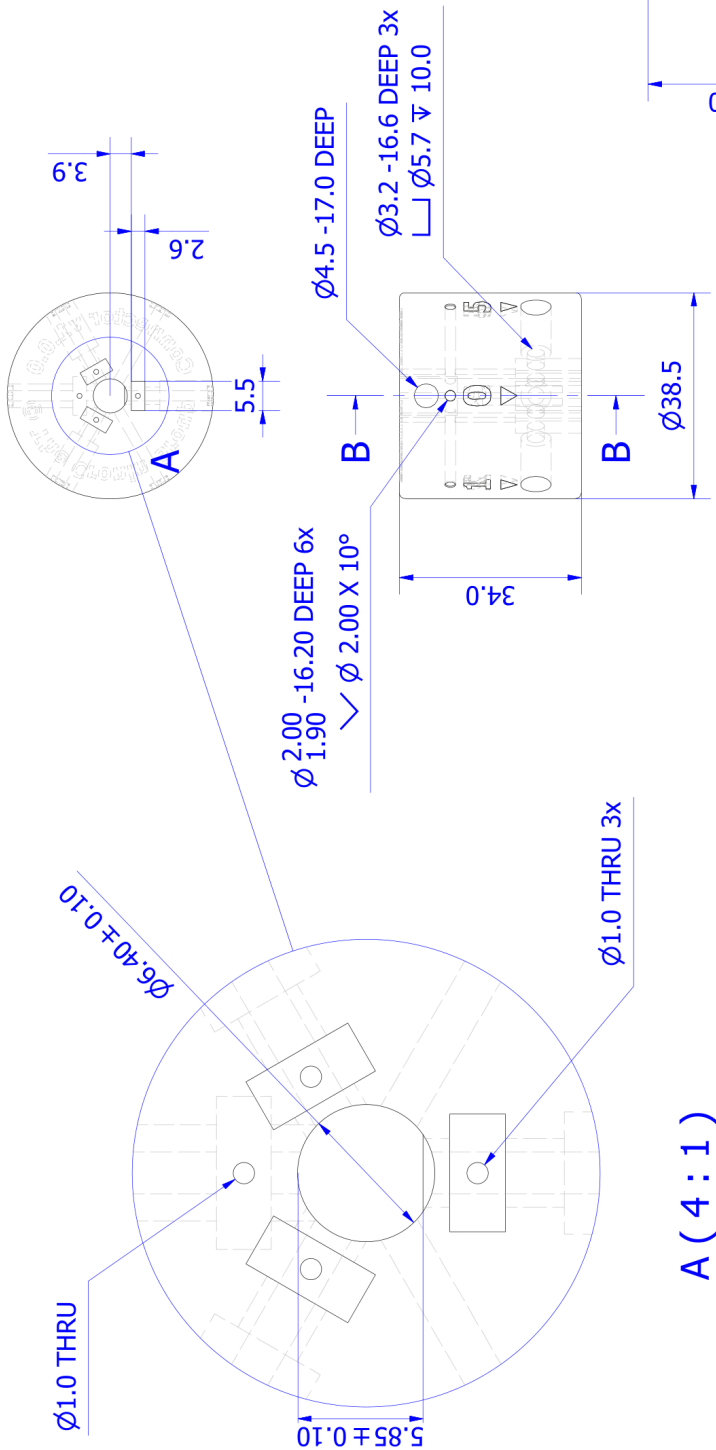
NOTES

- Outer edges should be broken by fillets or chamfers
- Part number, revision and copyright must be embossed on the top surface
- A 3D model is available

A

A

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm. CENTRE DISTANCE ±0.25mm. ANGLES 0°30".		DRAWN BY: Sebastian Steiner		REVIEWED BY: Graham Sharp		UNCONTROLLED COPY IF PRINTED			
TOLERANCE CLASS: TO SUPPLIER SPEC		CROWN GROUP COMPLEX CHEMICAL SYSTEMS		APPROVED BY: Lee Cronin		DO NOT SCALE: IF IN DOUBT, ASK			
MATERIAL: RDG720		© CRONIN GROUP UNIVERSITY OF GLASGOW		THIS WORK IS LICENCED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.		DESCRIPTION: Housing for Chemputer valve			
SPEC: Smooth - Ivory				DATE: 28/11/2017		PART NO: Valve Housing		VERSION: 1.0.0	
FINISH: MATTE				SHEET 1/1		SIZE A3		SCALE 1 : 1	
WEIGHT: 90.8 g									



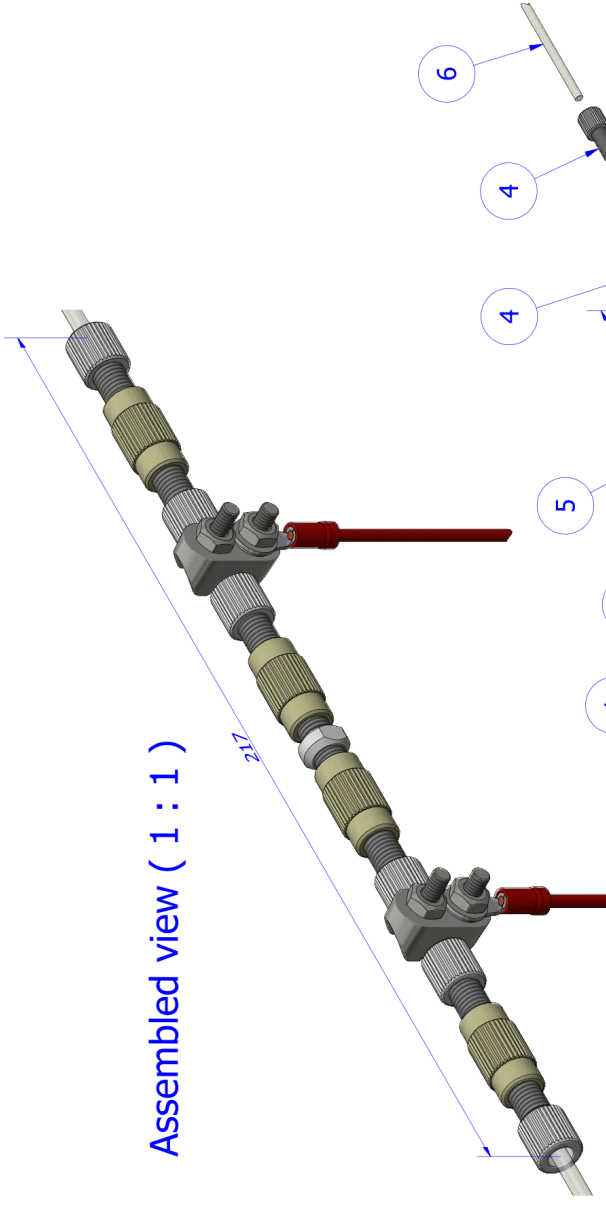
NOTES

1. Outer edges should be broken by fillets or chamfers
2. Part number, revision and copyright must be embossed on the top surface
3. Arrows and numbers denoting the current valve position should be painted, embossed or imprinted on the sides

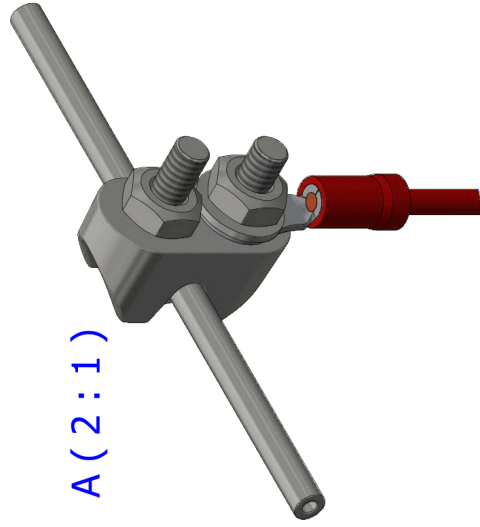
ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°-30°.	DRAWN BY: Sebastian Steiner	REVIEWED BY: Graham Sharp APPROVED BY: Lee Cronin	UNCONTROLLED COPY IF PRINTED DO NOT SCALE: IF IN DOUBT, ASK
	DESCRIPTION: Motor connector for Chemputer Valve		
TOLERANCE CLASS: TO SUPPLIER SPEC	THIS WORK IS LICENSED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.		
MATERIAL: RDGT20	DATE: 28/11/2017		
SPEC: Smooth - Ivory	SHEET: 1/1	SCALE: A3	PART NO.: 1
FINISH: MATTE	WEIGHT: 4.32 g	SIZE: 1 : 1	VERSION: 1.0.0
© CRONIN GROUP UNIVERSITY OF GLASGOW		SHEET: 2	PART NO.: 1

PARTS LIST		
ITEM	QTY	PART NUMBER DESCRIPTION
1	1	1/4-28 UNF union male Fluidic union 1/4-28 UNF male
2	4	1/4-28 UNF union female Fluidic union 1/4-28 UNF female
3	6	1/8" Ferrule Compression ferrule for 1/8" O.D. tubing
4	6	1/4-28 Fitting Flangeless fitting for 1/8" tubing
5	2	1/8" steel tubing section Section of 1/8" O.D. stainless steel tubing cut to length
6	2	1/8" PTFE tubing section Section of 1/8" O.D. clear PTFE tubing
7	2	Rope grip base DIN741 rope grip for 3mm rope, base
8	2	Rope grip hook DIN741 rope grip for 3mm rope, hook
9	2	Crimp Terminal Ring crimp terminal
10	2	DIN 125 - A 4.3 Washer
11	4	DIN 934 - M4 Hex Nut
12	2	Connection Wire 20 AWG equipment wire

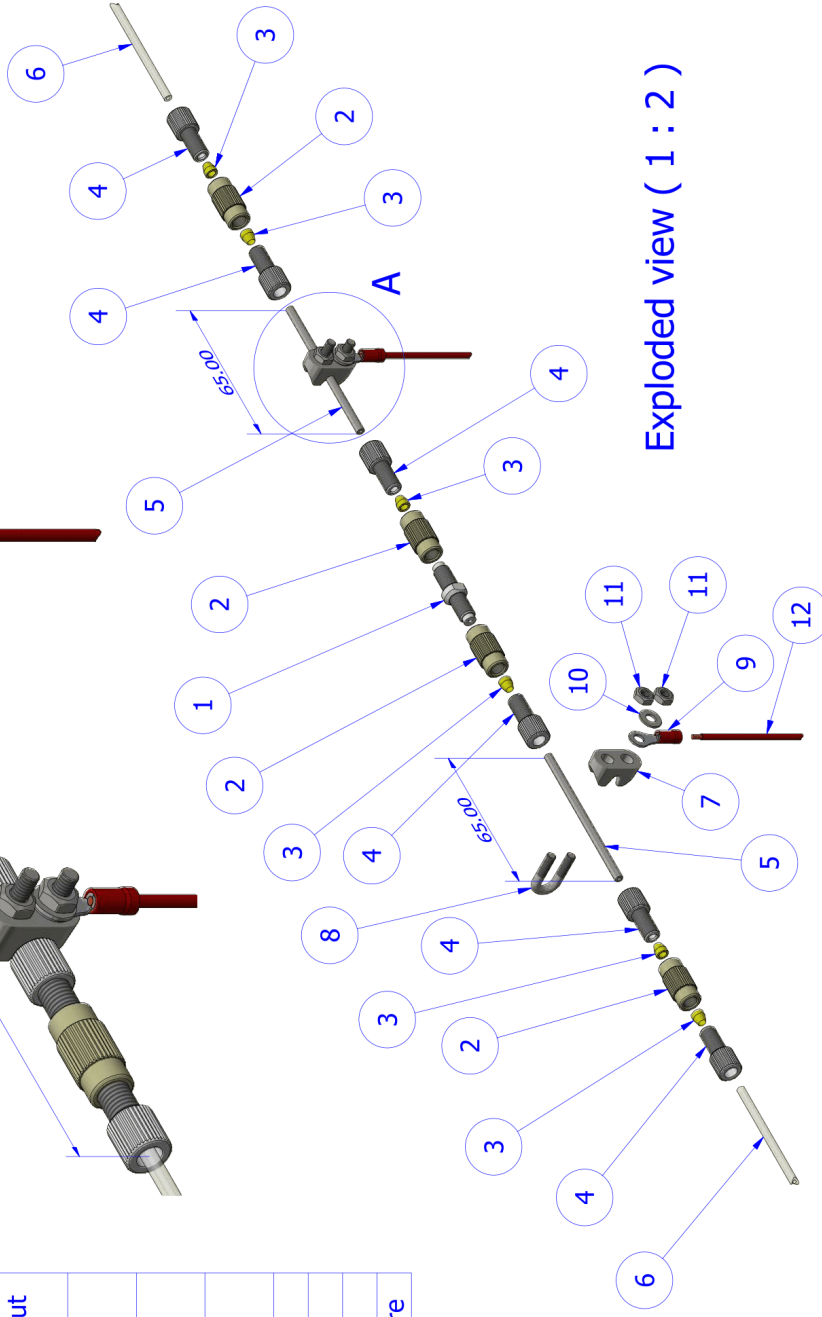
Assembled view (1 : 1)



A (2 : 1)



Exploded view (1 : 2)



ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°-30°.

DRAWN BY: Sebastian Steiner
 REVIEWED BY: Graham Sharp
 APPROVED BY: Lee Cronin

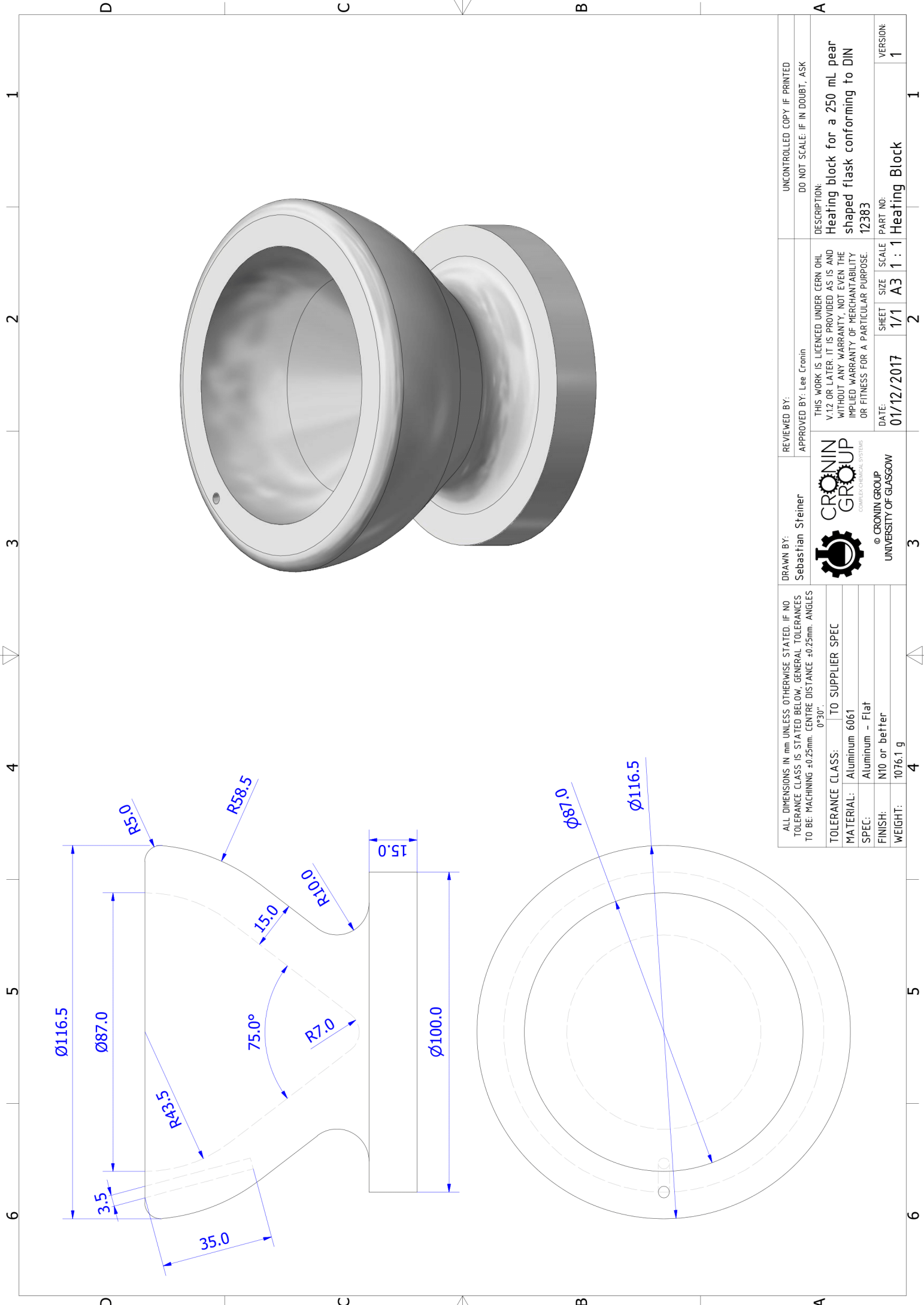


THIS WORK IS LICENSED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.


UNCONTROLLED COPY IF PRINTED
 DO NOT SCALE: IF IN DOUBT, ASK

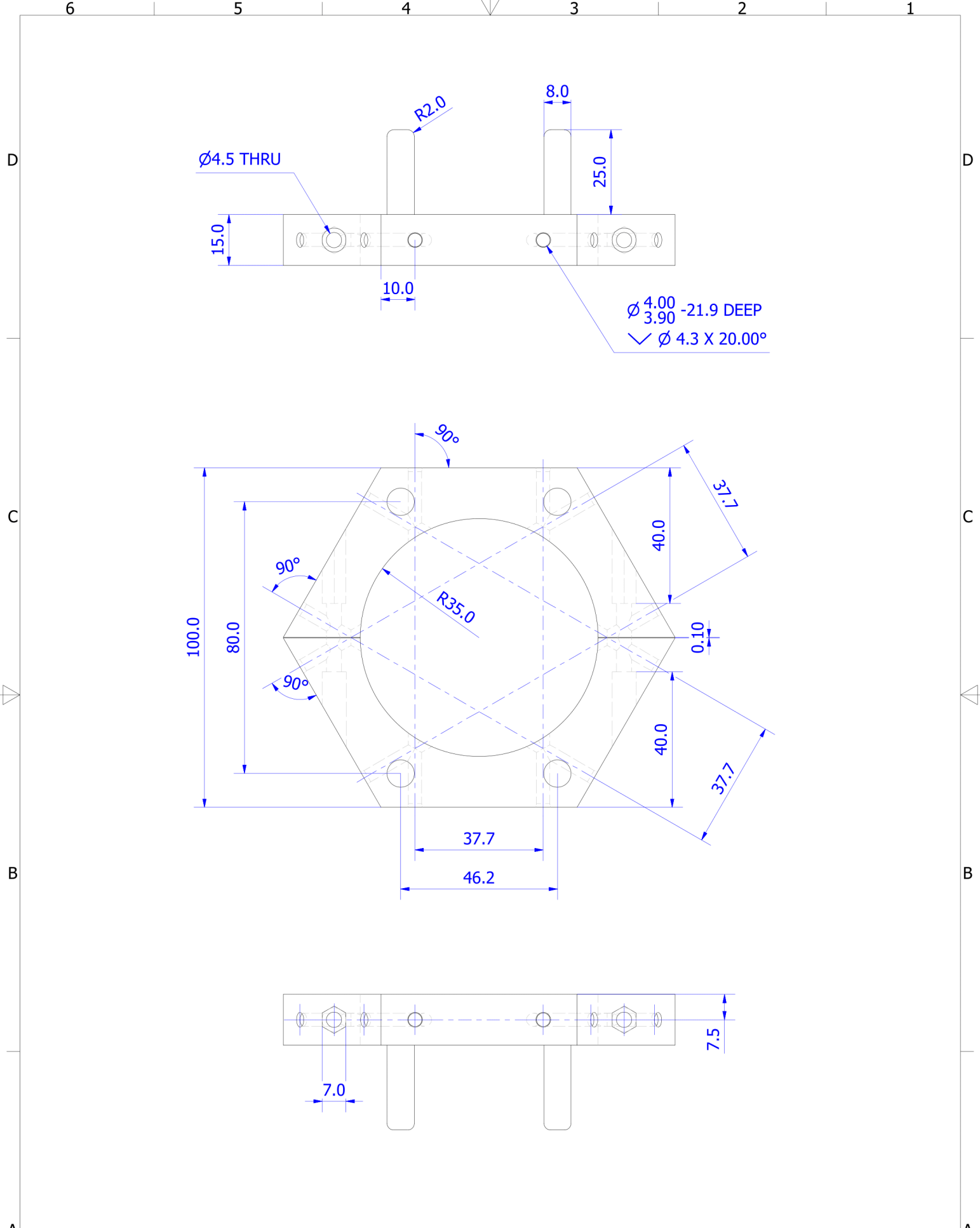
DESCRIPTION:
Flow conductivity sensor

TOLERANCE CLASS:	TO SUPPLIER SPEC
MATERIAL:	
SPEC:	
FINISH:	42.1 g
WEIGHT:	4
DATE:	29/11/2017
SHEET:	1/1
SCALE:	A3
SIZE:	1 : 2
PART NO.:	Chemputer Conductivity Sensor
VERSION:	1



ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°-30°.	
TOLERANCE CLASS:	TO SUPPLIER SPEC
MATERIAL:	Aluminum 6061
SPEC:	Aluminum - Flat
FINISH:	N10 or better
WEIGHT:	1076.1 g

DRAWN BY: Sebastian Steiner	REVIEWED BY: APPROVED BY: Lee Cronin	 © CRONIN GROUP UNIVERSITY OF GLASGOW	UNCONTROLLED COPY IF PRINTED DO NOT SCALE: IF IN DOUBT, ASK
			DESCRIPTION: Heating block for a 250 mL pear shaped flask conforming to DIN 12383
DATE: 01/12/2017		SHEET: 1/1	SCALE: A3
PART NO: 12383		SHEET: 2	SCALE: 1 : 1
Heating Block		VERSION: 1	1



ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm. CENTRE DISTANCE ±0.25mm. ANGLES 0°30'.	
TOLERANCE CLASS:	TO SUPPLIER SPEC
MATERIAL:	RDG720
SPEC:	Smooth - Ivory
FINISH:	GLOSSY
WEIGHT:	84.9 g

DRAWN BY:
Sebastian Steiner



© CRONIN GROUP
UNIVERSITY OF GLASGOW

REVIEWED BY: Graham Sharp
APPROVED BY: Lee Cronin

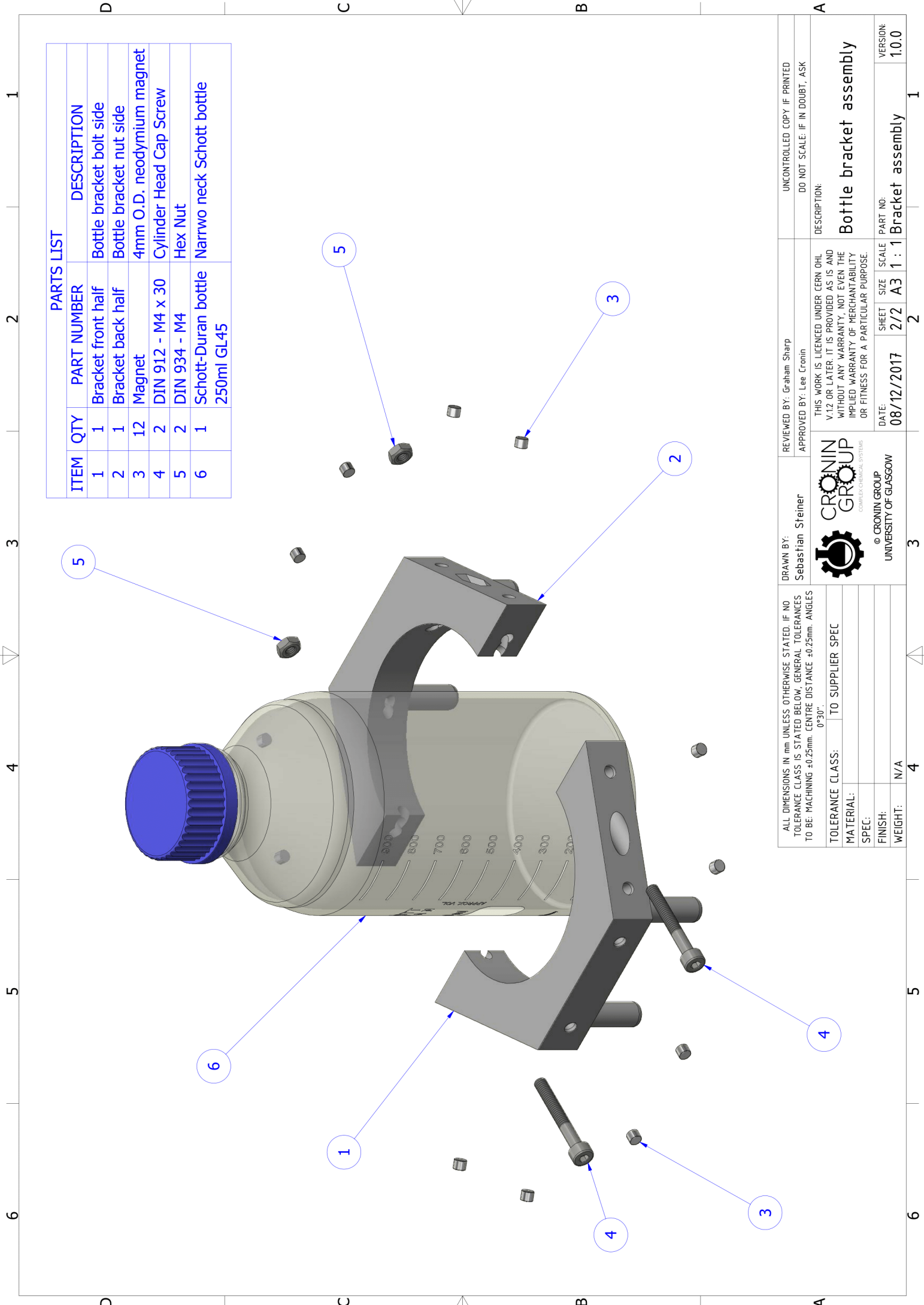
THIS WORK IS LICENCED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

DATE:	SHEET	SIZE	SCALE
08/12/2017	1/2	A3	1 : 1

UNCONTROLLED COPY IF PRINTED
DO NOT SCALE: IF IN DOUBT, ASK

DESCRIPTION:
Modular rack bracket for 250mL Schott bottles

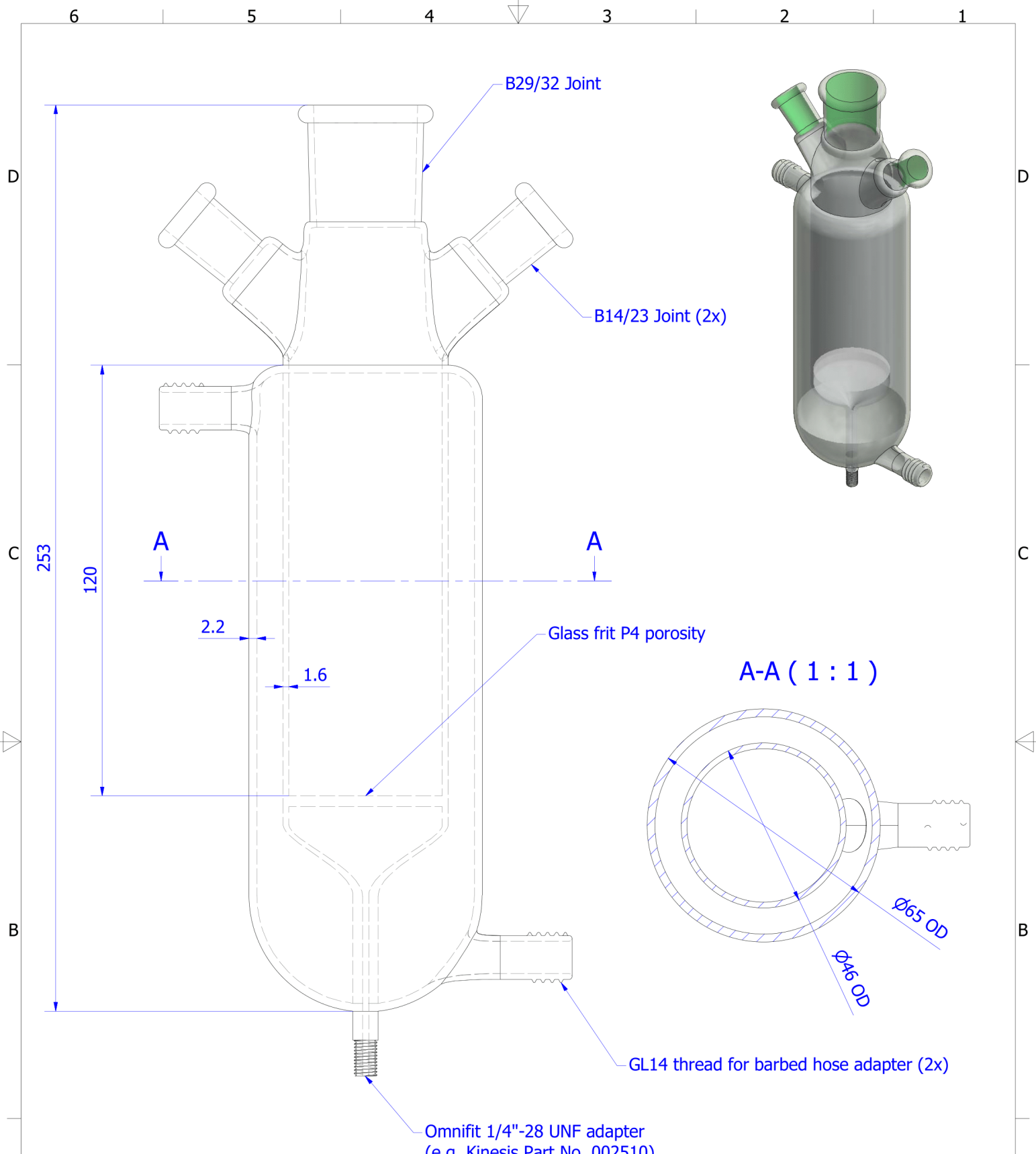
PART NO:	VERSION:
Chemputer Bottle Bracket	1.0.0



PARTS LIST


ITEM	QTY	PART NUMBER	DESCRIPTION
1	1	Bracket front half	Bottle bracket bolt side
2	1	Bracket back half	Bottle bracket nut side
3	12	Magnet	4mm O.D. neodymium magnet
4	2	DIN 912 - M4 x 30	Cylinder Head Cap Screw
5	2	DIN 934 - M4	Hex Nut
6	1	Schott-Duran bottle 250ml GL45	Narrow neck Schott bottle

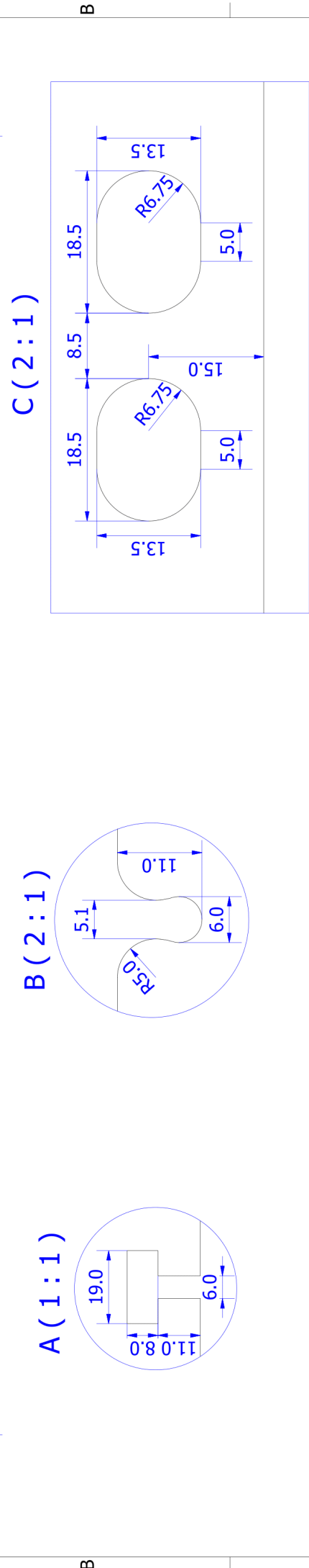
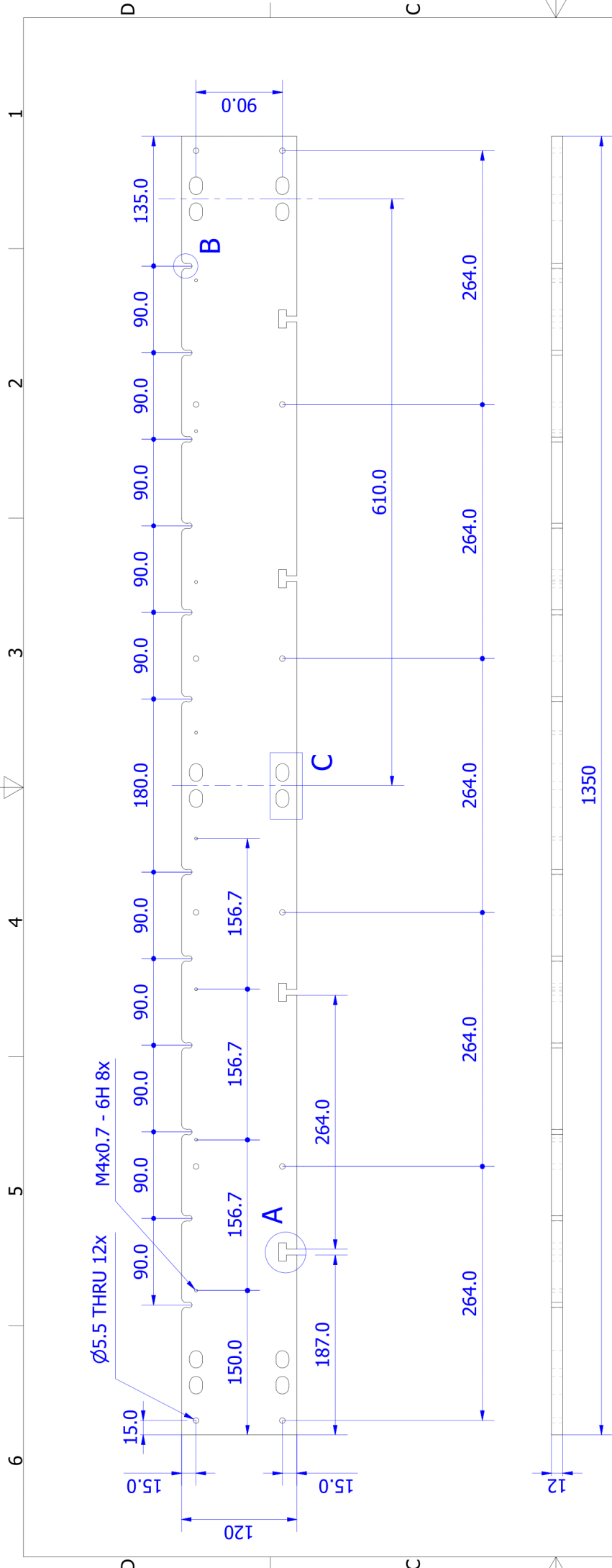
<p>ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm. CENTRE DISTANCE ±0.25mm. ANGLES 0°-30".</p> <p>TOLERANCE CLASS: TO SUPPLIER SPEC</p> <p>MATERIAL:</p> <p>SPEC:</p> <p>FINISH: N/A</p> <p>WEIGHT: 4</p>	<p>DRAWN BY: Sebastian Steiner</p> <p>REVIEWED BY: Graham Sharp</p> <p>APPROVED BY: Lee Cronin</p> <p>THIS WORK IS LICENSED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.</p> <p>DATE: 08/12/2017</p> <p>SHEET: 2/2</p> <p>SCALE: A3</p> <p>SIZE: 1 : 1</p>	<p>UNCONTROLLED COPY IF PRINTED</p> <p>DO NOT SCALE: IF IN DOUBT, ASK</p> <p>DESCRIPTION:</p> <p>Bottle bracket assembly</p> <p>PART NO: Bracket assembly</p> <p>VERSION: 1.0.0</p>
---	---	--



NOTES


1. Omnifit adapter can be supplied
2. There are no critical dimensions, all measurements are approximate
3. Dead volume under the frit should be minimised

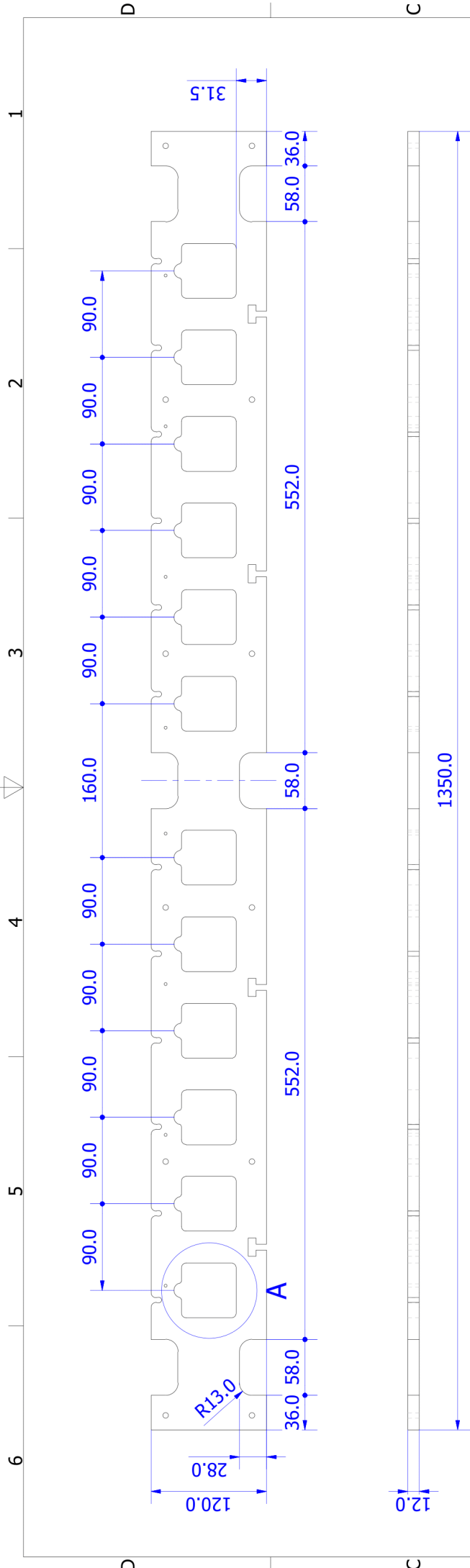
ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm. CENTRE DISTANCE ±0.25mm. ANGLES 0°30".		DRAWN BY: Sebastian Steiner		REVIEWED BY: John Liddell APPROVED BY: Lee Cronin		UNCONTROLLED COPY IF PRINTED DO NOT SCALE: IF IN DOUBT, ASK		
TOLERANCE CLASS: NO CRITICAL DIMENSIONS		 <p>© CRONIN GROUP UNIVERSITY OF GLASGOW</p>		<p>THIS WORK IS LICENCED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.</p>		<p>DESCRIPTION: Jacketed filter with UNF connector</p>		
MATERIAL:	Glass							
SPEC:	Clear							
FINISH:								
WEIGHT:	307.4 g	DATE:	08/03/2018	SIZE:	A3	SCALE:	1 : 1	
				PART NO:	Chemputer Jacketed Filter		VERSION:	1.0.0



NOTES

- The part is symmetrical around the vertical axis

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ± 0.25 mm. CENTRE DISTANCE ± 0.25 mm. ANGLES $0^{\circ}30'$.	REVIEWED BY: Jim McIver APPROVED BY: Lee Cronin	UNCONTROLLED COPY IF PRINTED DO NOT SCALE: IF IN DOUBT, ASK
TOLERANCE CLASS: TO SUPPLIER SPEC	DRAWN BY: Sebastian Steiner	DESCRIPTION: Bottom half of a shelf for either pumps or valves
MATERIAL: Polypropylene	 CRONIN GROUP CONSULTING ENGINEERS	THIS WORK IS LICENSED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.
SPEC: Smooth - Ash	© CRONIN GROUP UNIVERSITY OF GLASGOW	DATE: 14/05/2018
FINISH:	WEIGHT: 1.7 kg	PART NO: Shelf bottom
		SCALE: A3 1 : 4
		VERSION: 0.9.0

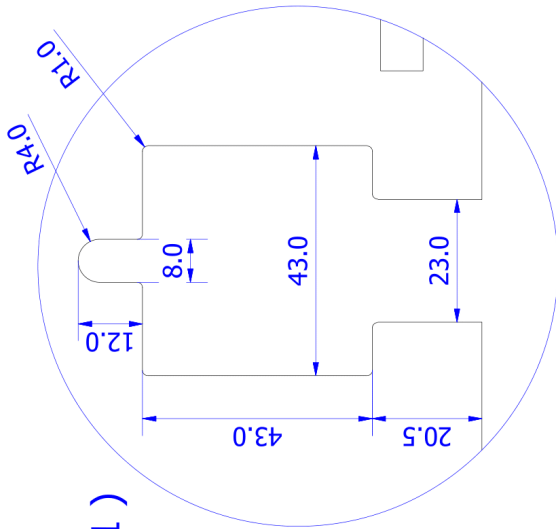
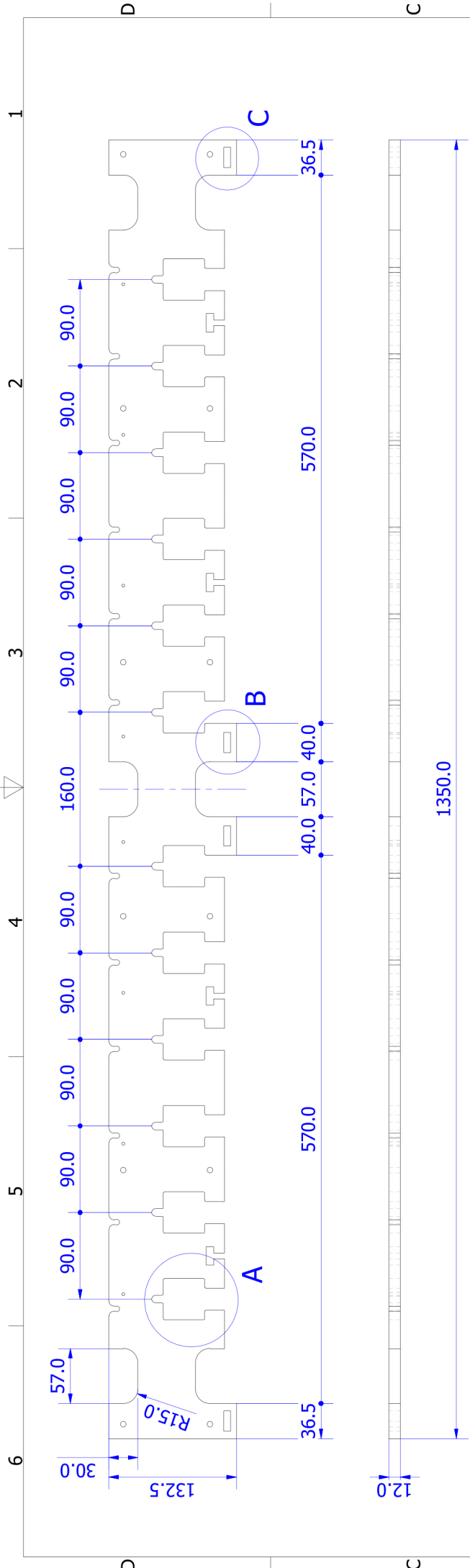


A (1 : 1)

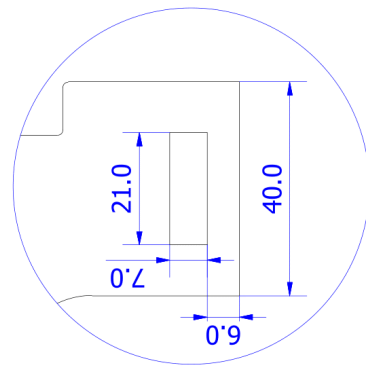
NOTES

1. The part is symmetrical around the vertical axis.
2. The part is derived from the shelf bottom. Any dimensions not specified in this drawing can be found in the drawing of the parent part.
3. Detail A is indicative of the other occurrences of the respective features.

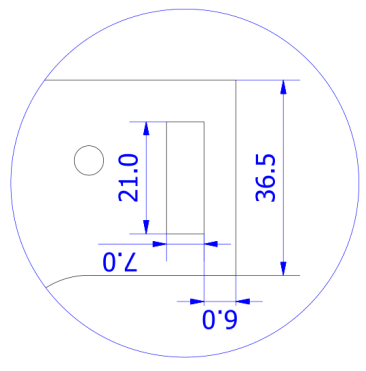
ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm, CENTRE DISTANCE ±0.25mm, ANGLES 0°-30°.	DRAWN BY: Sebastian Steiner	REVIEWED BY: Jim McIver APPROVED BY: Lee Cronin	UNCONTROLLED COPY IF PRINTED DO NOT SCALE: IF IN DOUBT, ASK
	TOLERANCE CLASS: TO SUPPLIER SPEC MATERIAL: Polypropylene SPEC: Smooth - Ash		DESCRIPTION: Top half of a shelf for valves
FINISH: 12 kg	DATE: 14/05/2018	SCALE: A3 1 : 4	PART NO: Shelf top for valves
WEIGHT: 1.2 kg	© CRONIN GROUP UNIVERSITY OF GLASGOW	SIZE: A3	VERSION: 0.9.0



A (1 : 1)




B (1 : 1)

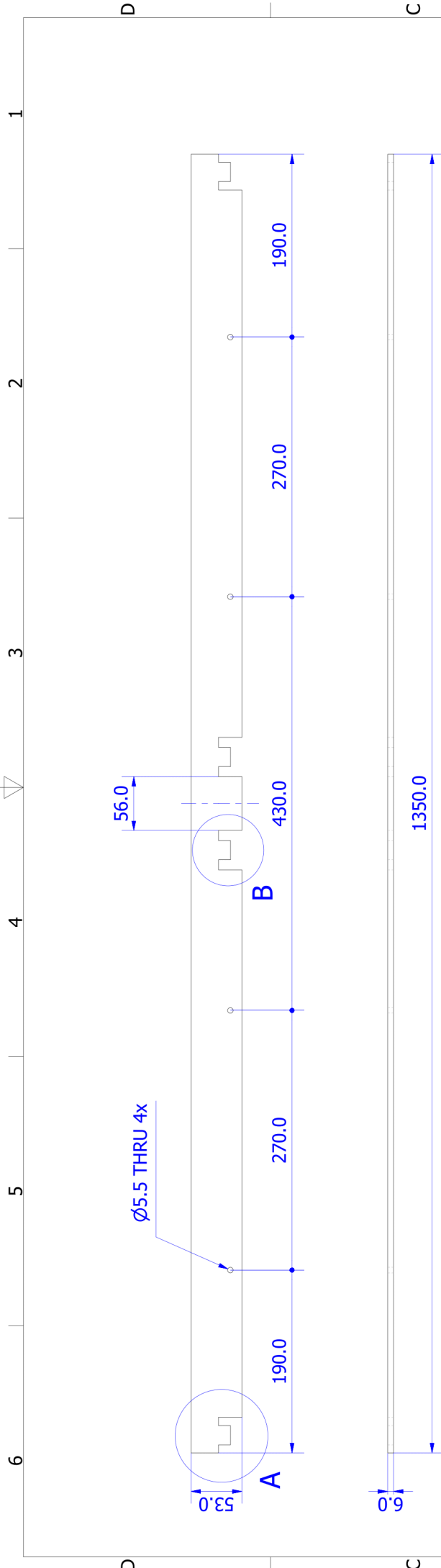


C (1 : 1)

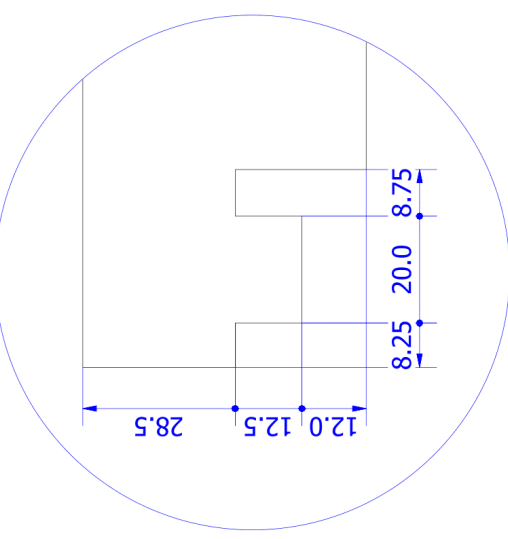
NOTES

1. The part is symmetrical around the vertical axis.
2. The part is derived from the shelf bottom. Any dimensions not specified in this drawing can be found in the drawing of the parent part.
3. Details A, B and C are indicative of the other occurrences of the respective features.

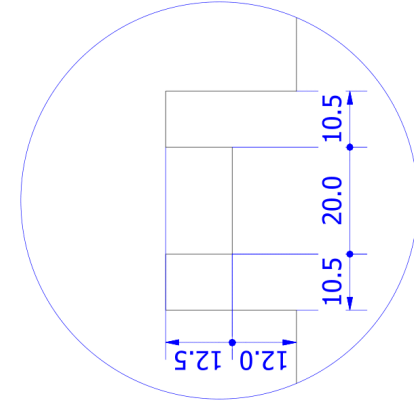
DRAWN BY: Sebastian Steiner	REVIEWED BY: Jim McIver	UNCONTROLLED COPY IF PRINTED
	APPROVED BY: Lee Cronin	DO NOT SCALE: IF IN DOUBT, ASK
 © CRONIN GROUP UNIVERSITY OF GLASGOW	THIS WORK IS LICENSED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.	DESCRIPTION: Top half of a shelf for pumps
	DATE: 14/05/2018	PART NO: A3 1 : 4 Shelf top for pumps
ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ±0.25mm; CENTRE DISTANCE ±0.25mm; ANGLES 0°-30".	TOLERANCE CLASS: TO SUPPLIER SPEC	
	MATERIAL: Polypropylene	
	SPEC: Smooth - Ash	
	FINISH: 1.3 kg	
	WEIGHT: 1.3 kg	



A(1:1)



B(1:1)



NOTES

- The part is symmetrical around the vertical axis.

ALL DIMENSIONS IN mm UNLESS OTHERWISE STATED. IF NO TOLERANCE CLASS IS STATED BELOW, GENERAL TOLERANCES TO BE: MACHINING ± 0.25 mm. CENTRE DISTANCE ± 0.25 mm. ANGLES $0^{\circ}30'$.

TOLERANCE CLASS:	TO SUPPLIER SPEC
MATERIAL:	Polypropylene
SPEC:	Smooth - Ash
FINISH:	
WEIGHT:	0.4 kg

DRAWN BY:
Sebastian Steiner



© CRONIN GROUP
UNIVERSITY OF GLASGOW

REVIEWED BY: Jim McIver
APPROVED BY: Lee Cronin

THIS WORK IS LICENSED UNDER CERN OHL V.1.2 OR LATER. IT IS PROVIDED AS IS AND WITHOUT ANY WARRANTY, NOT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

DATE:
14/05/2018

SCALE
A3 1 : 4

PART NO:
1 : 4 Pump shelf front

VERSION:
0.9.0

UNCONTROLLED COPY IF PRINTED
DO NOT SCALE: IF IN DOUBT, ASK

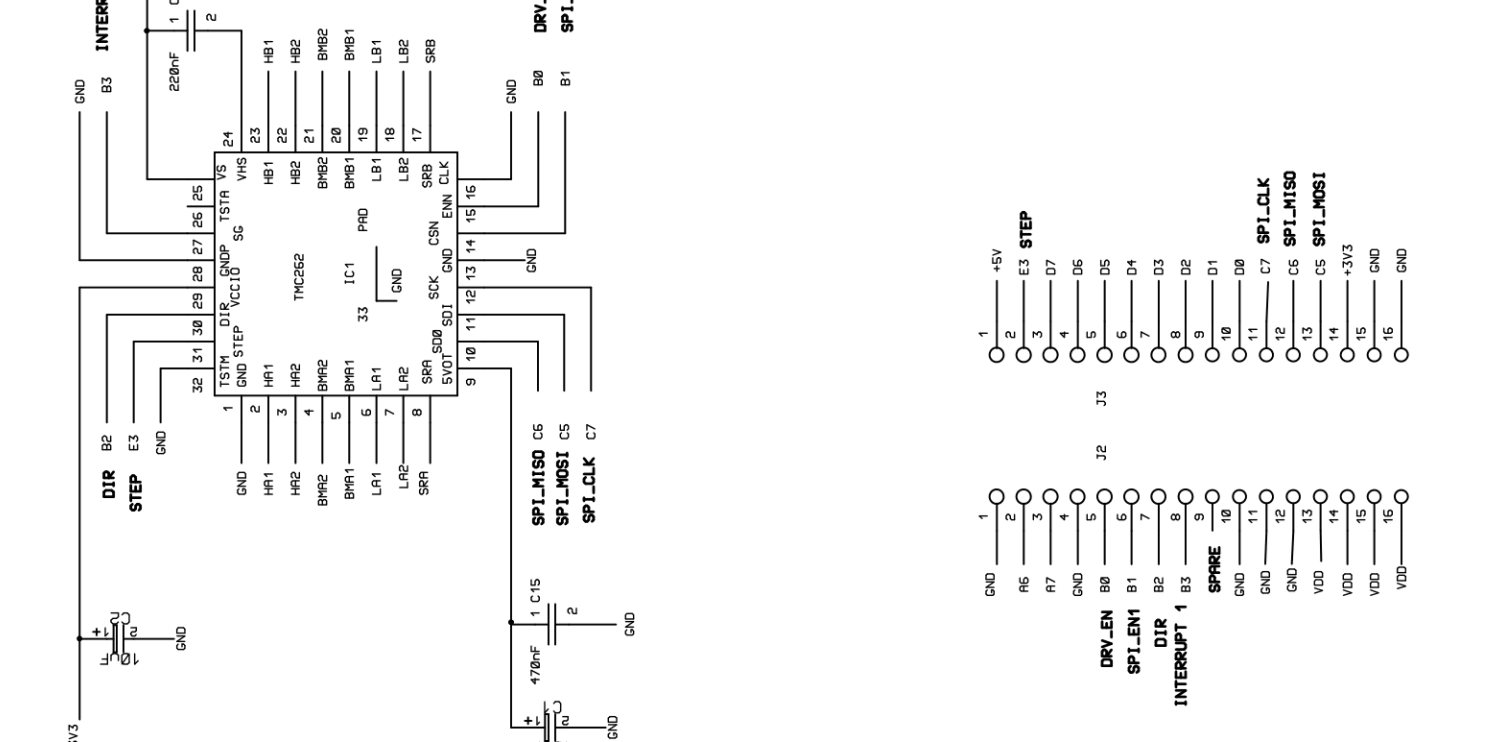
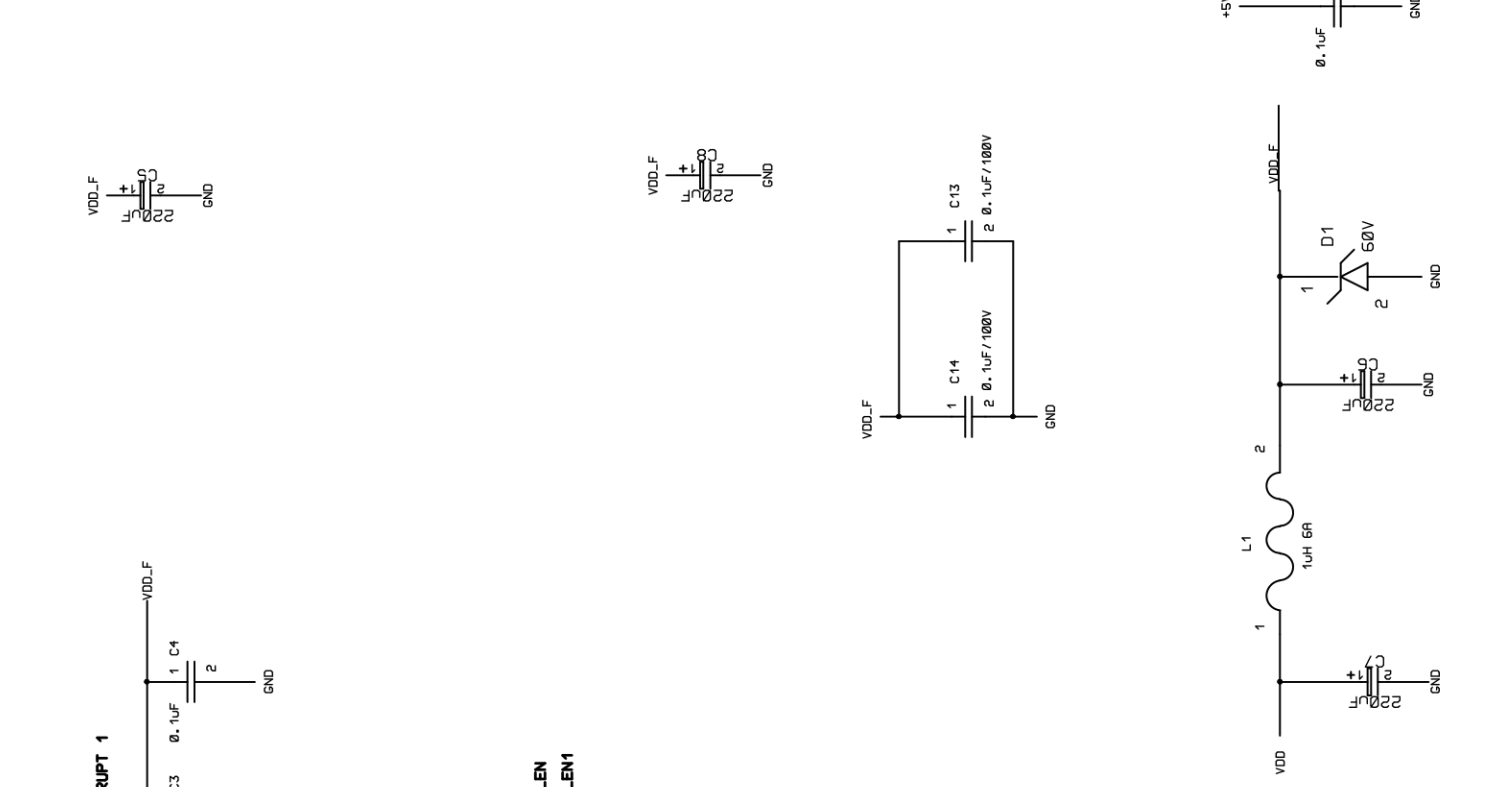
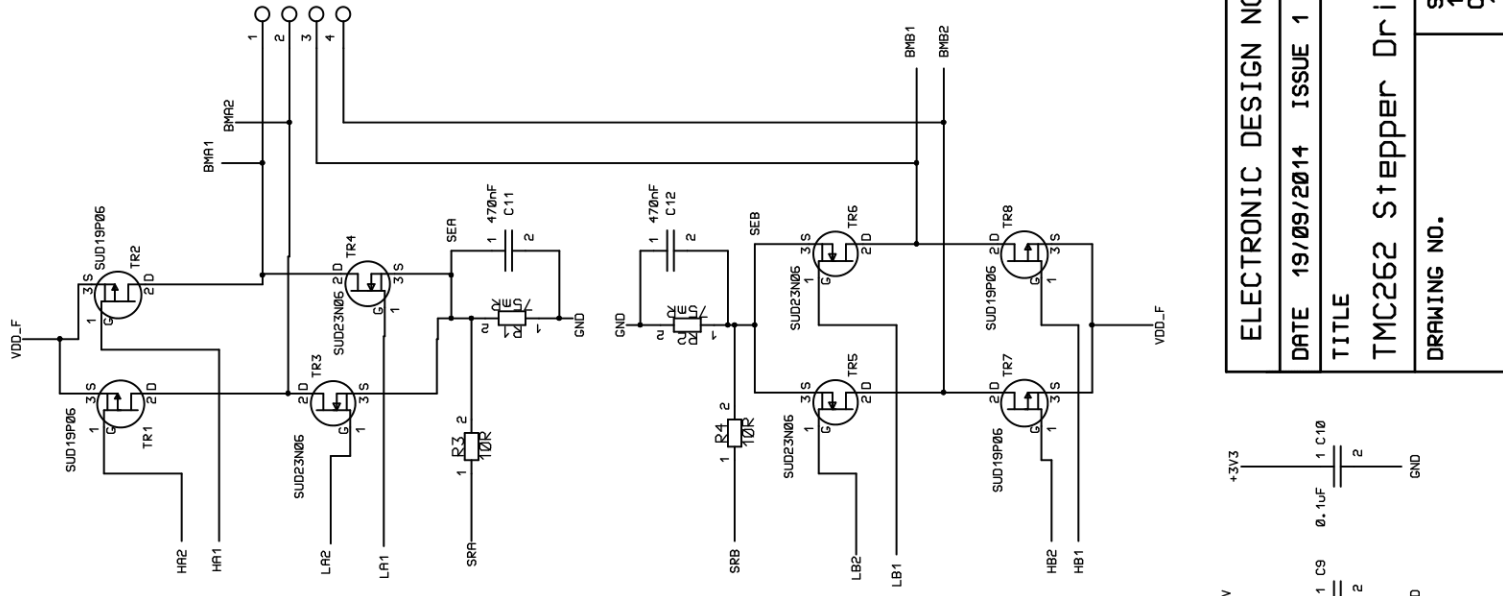
DESCRIPTION:
Front panel for a pump rack, preventing pumps from topping

APPENDIX

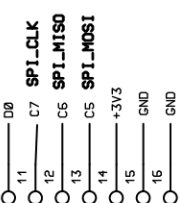
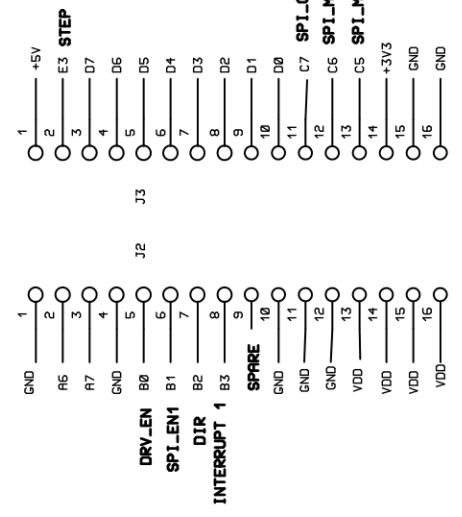
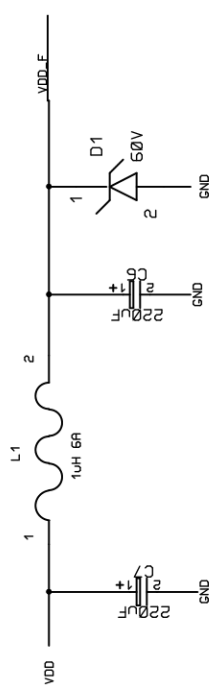
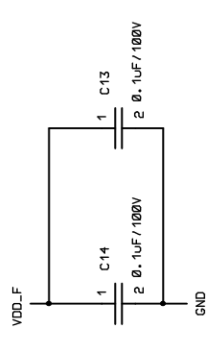
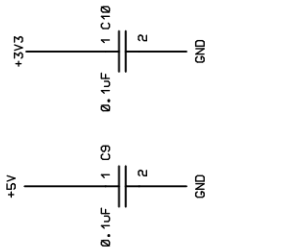
III CONTROL PCB SCHEMATICS

NOTA BENE: Control boards were designed by an external contractor. The Cronin Group holds all copyrights. The following schematics are enclosed for information only and DO NOT constitute original work performed for this thesis!

APPENDIX



ELECTRONIC DESIGN NOW	
DATE 19/09/2014	ISSUE 1
TITLE	
TMC262 Stepper Drive	
DRAWING NO.	
SHT	1 OF 1



IV CHASM REFERENCE

Command arguments for the Chemical Assembly Language have the following structure:

- {CMD}({ARGS});

All commands are capitalised. Arguments are in parentheses and separated by commas. A line is terminated by a semicolon. See below for a full list of supported commands.

IV.I.I PUMPS AND VALVES

NOTE The pump names are the names as defined in the GraphML definition of the platform.

Command: MOVE ({src}, {dest}, {volume}, {move_speed}, {aspiration_speed}, {dispense_speed});

Moves a specified volume from one node in the graph to another. Moving from and to the same node is supported.

- **{src}** is the name of the source flask
- **{dest}** is the name of the destination flask
- **{volume}** can be a float or "all" in which case it moves the entire current volume
- **{move_speed}** is the speed at which it moves material across the Backbone. This argument is optional, if absent, it defaults to 50mL/min
- **{aspiration_speed}** is the speed at which it aspirates from the source. This argument is optional, if absent, it defaults to {move_speed}. It will only be parsed as {aspiration_speed} if a move speed is given (argument is positional)
- **{dispense_speed}** is the speed at which it aspirates from the source. This argument is optional, if absent, it defaults to {move_speed}. It will only be parsed as {aspiration_speed} if a move speed and aspiration speed is given (argument is positional)

Command: HOME ({pump_name}, {move_speed});

Moves a given pump to home.

- **{pump_name}** is the name of the pump to be homed.
- **{move_speed}** is the requested speed in mL/min.

Command: SEPARATE ({lower_phase_target}, {upper_phase_target});

Launches a phase separation sequence. The name of the separator is currently hard-coded!

- **{lower_phase_target}** is the name of the flask the lower phase should be transferred to.
- **{upper_phase_target}** is the name of the flask the upper phase should be transferred to. If "separator_top" is specified, the upper phase is left in the separator.

Command: PRIME ({aspiration_speed});

Moves the tube volume of every node with "flask" as class to waste.

- **{aspiration_speed}** is the speed in mL/min at which material should be withdrawn.

Command: SWITCH_VACUUM ({flask}, {destination});

Switches a vacuum valve between Backbone and vacuum.

- **{flask}** is the name of the node the vacuum valve is logically attached to (e.g. "filter_bottom")
- **{destination}** either "vacuum" or "Backbone"

Command: SWITCH_CARTRIDGE ({flask}, {cartridge});

Switches a cartridge carousel to the specified position.

- **{flask}** is the name of the node the vacuum valve is logically attached to (e.g. "rotavap")
- **{cartridge}** is the number of the position the carousel should be switched to (0-5)

Command: SWITCH_COLUMN ({column}, {destination});

Switches a fractionating valve attached to a chromatography column.

- **{column}** is the name of the column in the graph
- **{destination}** either "collect" or "waste"

IV.I.II STIRRER PLATES AND OVERHEAD STIRRERS

NOTE The parameter {name} refers to the node the device is attached to (e.g. reactor_reactor)

Command: START_STIR ({name});

Starts the stirring operation of a hotplate or overhead stirrer.

- **{name}** is the name of the node the device is attached to.

Command: START_HEAT ({name});

Starts the stirring operation of a hotplate stirrer. This command is NOT available for overhead stirrers.

- **{name}** is the name of the node the device is attached to.

Command: STOP_STIR ({name});

Stops the stirring operation of a hotplate or overhead stirrer.

- **{name}** is the name of the node the device is attached to.

Command: STOP_HEAT ({name});

Stops the stirring operation of a hotplate stirrer. This command is NOT available for overhead stirrers.

- **{name}** is the name of the node the device is attached to.

Command: SET_TEMP ({name}, {temp});

Sets the temperature setpoint of a hotplate stirrer. This command is NOT available for overhead stirrers.

- **{name}** is the name of the node the device is attached to.
- **{temp}** is the required temperature in °C

Command: SET_STIR_RPM ({name}, {rpm});

Sets the stirring speed setpoint of a hotplate or overhead stirrer.

- **{name}** is the name of the node the device is attached to.
- **{rpm}** is the speed setpoint in rpm.

Command: STIRRER_WAIT_FOR_TEMP ({name});

Delays the script execution until the current temperature of the hotplate is within 0.5°C of the setpoint. This command is NOT available for overhead stirrers.

- **{name}** is the name of the node the device is attached to.

IV.I.III ROTARY EVAPORATOR

NOTE The parameter {name} refers to the node representing the top flask of the roti (e.g. rotavap)

Command: START_HEATER_BATH ({name});

Starts the heating bath of a rotary evaporator.

- **{name}** is the name of the node representing the rotary evaporator.

Command: STOP_HEATER_BATH ({name});

Stops the heating bath of a rotary evaporator.

- **{name}** is the name of the node representing the rotary evaporator.

Command: START_ROTATION ({name});

Starts the rotation of a rotary evaporator.

- **{name}** is the name of the node representing the rotary evaporator.

Command: STOP_ROTATION ({name});

Stops the rotation of a rotary evaporator.

- **{name}** is the name of the node representing the rotary evaporator.

Command: LIFT_ARM_UP ({name});

Lifts the rotary evaporator up.

- **{name}** is the name of the node representing the rotary evaporator.

Command: LIFT_ARM_DOWN ({name});

Lifts the rotary evaporator down.

- **{name}** is the name of the node representing the rotary evaporator.

Command: RESET_ROTAVAP ({name});

Resets the rotary evaporator.

- **{name}** is the name of the node representing the rotary evaporator.

Command: SET_BATH_TEMP ({name}, {temp});

Sets the temperature setpoint for the heating bath.

- **{name}** is the name of the node representing the rotary evaporator.
- **{temp}** is the temperature setpoint in °C.

Command: SET_ROTATION ({name}, {rotation});

Sets the rotation speed setpoint for the rotary evaporator.

- **{name}** is the name of the node representing the rotary evaporator.
- **{rotation}** is the speed setpoint in rpm.

Command: RV_WAIT_FOR_TEMP ({name});

Delays the script execution until the current temperature of the heating bath is within 0.5°C of the setpoint.

- **{name}** is the name of the node representing the rotary evaporator.

Command: SET_INTERVAL ({name}, {interval});

Sets the interval time for the rotary evaporator, causing it to periodically switch direction. Setting this to 0 deactivates interval operation.

- **{name}** is the name of the node representing the rotary evaporator.
- **{interval}** is the interval time in seconds.

IV.I.IV VACUUM PUMP

NOTE The parameter {name} refers to the node the device is attached to (e.g. rotavap)

Command: INIT_VAC_PUMP ({name});

Initialises the vacuum pump controller.

- **{name}** is the name of the node the vacuum pump is attached to.

Command: GET_VAC_SP ({name});

Reads the current vacuum setpoint.

- **{name}** is the name of the node the vacuum pump is attached to.

Command: SET_VAC_SP ({name}, {set_point});

Sets a new vacuum setpoint.

- **{name}** is the name of the node the vacuum pump is attached to.
- **{set_point}** is the vacuum setpoint in mbar.

Command: START_VAC ({name});

Starts the vacuum pump.

- **{name}** is the name of the node the vacuum pump is attached to.

Command: STOP_VAC ({name});

Stops the vacuum pump.

- **{name}** is the name of the node the vacuum pump is attached to.

Command: VENT_VAC ({name});

Vents the vacuum pump to ambient pressure.

- **{name}** is the name of the node the vacuum pump is attached to.

Command: SET_SPEED_SP ({name}, {set_point});

Sets the speed of the vacuum pump (0-100%).

- **{name}** is the name of the node the vacuum pump is attached to.
- **{set_point}** is the vacuum pump speed in percent.

IV.I.V RECIRCULATION CHILLER

NOTE The parameter {name} refers to the node the device is attached to (e.g. rotavap)

Command: START_CHILLER ({name});

Starts the recirculation chiller.

- **{name}** is the name of the node the chiller is attached to.

Command: STOP_CHILLER ({name});

Stops the recirculation chiller.

- **{name}** is the name of the node the chiller is attached to.

Command: SET_CHILLER ({name}, {setpoint});

Sets the temperature setpoint.

- **{name}** is the name of the node the chiller is attached to.
- **{setpoint}** is the temperature setpoint in °C.

Command: CHILLER_WAIT_FOR_TEMP ({name});

Delays the script execution until the current temperature of the chiller is within 0.5°C of the setpoint.

- **{name}** is the name of the node the chiller is attached to.

Command: RAMP_CHILLER ({name}, {ramp_duration}, {end_temperature});

Causes the chiller to ramp the temperature up or down. Only available for Petite Fleur.

- **{name}** is the name of the node the chiller is attached to.
- **{ramp_duration}** is the desired duration of the ramp in seconds.
- **{end_temperature}** is the final temperature of the ramp in °C.

Command: SWITCH_CHILLER ({name}, {state});

Switches the solenoid valve.

- **{name}** is the name of the node the solenoid valve is attached to.
- **{state}** is either "on" or "off"

Command: SET_COOLING_POWER ({name}; {cooling_power});

Sets the cooling power (0-100%). Only available for CF41.

- **{name}** is the name of the node the chiller is attached to.
- **{cooling_power}** is the desired cooling power in percent.

IV.I.VI CAMERA

Command: SET_RECORDING_SPEED ({speed});

Sets the time-lapse speed of the camera module.

- **{speed}** is the factor by which the recording should be sped up, i.e. 2 would mean twice the normal speed. 1 means normal speed.

IV.I.VII OTHER

Command: WAIT ({time});

Delays execution of the script for a set amount of time. This command will immediately reply with an estimate of when the waiting will be finished, and give regular updates indicating that it is still alive.

- **{time}** is the wait time in seconds.

Command: BREAKPOINT ({});

Introduces a breakpoint in the script. The execution is halted until the operator resumes it.

- **{}** Breakpoints take no arguments.