

5-axis 3D Printer

Designing a 5-axis 3D printer

Øyvind Kallevik Grutle
Master's Thesis Autumn 2015



5-axis 3D Printer

Øyvind Kallevik Grutle

3rd August 2015

Abstract

3D printers have in recent years become extremely popular. Even though 3D printing technology have existed since the late 1980's, it is now considered one of the most significant technological breakthroughs of the twenty-first century. Several different 3D printing processes have been invented during the years. But it is the fused deposition modeling (FDM) which was one of the first invented that is considered the most popular today. Even though the FDM process is the most popular, it still suffers from some issues.

This thesis looks at the possibility of removing these issues by using a 5-axis system. To explore this, a 5-axis FDM printer called the Pentarod have been created by extending a 3-axis FDM printer with two extra axes. Then parts that can simulate the two main issues of a FDM printer have been printed with the Pentarod to see if it can eliminate the lacks of a 3-axis FDM printer.

The Pentarod have been made successfully, and it have been able to print the parts without problems. By managing this Pentarod have shown that by adding two extra axes to a FDM printer, the two main issues for the FDM process can be eliminated.

Acknowledgements

I would like to thank my supervisor, Mats Høvin, for the continuous guidance and support throughout the thesis work.

I would also like to thank my fellow students, friends, family, and especially my significant other, Ingebjørg Johannessen, for the endless support I have received during this thesis.

Contents

I Introduction	1
1 Introduction	3
1.1 Introduction	3
1.2 Problems with FDM printers	3
1.3 Possible solutions	4
1.4 Goals of the thesis	4
1.5 Outline of the thesis	5
2 Background	7
2.1 Computer Numerical Control (CNC)	7
2.2 Additive Manufacturing	7
2.2.1 FDM printers	9
2.3 Subtractive Manufacturing	10
2.3.1 Multi-axis CNC mill	10
2.4 Tool path generation	11
2.5 G-code	12
2.5.1 5-axis G-code	12
2.6 Backlash	13
2.7 Open source	14
2.8 The RepRap project	14
2.9 Previous work	14
2.10 Tools and programs used	15
2.10.1 Duet	15
2.10.2 Stepper motors	16
2.10.3 Worm drive	17
2.10.4 Fortus 250mc	17
2.10.5 Objet Connex 500	17
2.10.6 Epilog Zing 24 Laser 30W	18
2.10.7 SolidWorks	18
2.10.8 HSMWorks	18
2.10.9 Printrun	18
2.10.10 G-code generators	19
2.10.11 C++	19
2.10.12 Java	19

II The Project	21
3 Building a 5-axis system	23
3.1 Planning the 5-axis system	23
3.1.1 Choosing a base system	23
3.1.2 Choosing 5-axis configuration	24
3.1.3 Finding the origin in the system	24
3.1.4 Choosing stepper motors	25
3.1.5 Controlling the stepper motors	25
3.1.6 Accuracy for the rotary system	26
3.2 Planning the first version	27
3.3 Modeling the first version	30
3.3.1 The platform	30
3.3.2 The rotary system	30
3.3.3 Lowering the nozzle	33
3.3.4 The Z-axis	33
3.4 Modeling result of the first version	35
3.5 Testing the first version	36
3.6 Problems with the first version	37
3.7 Planning the second version	37
3.8 Modeling the second version	38
3.8.1 The motor mount	38
3.8.2 The rotary system	39
3.8.3 The platform	40
3.8.4 Lowering the nozzle even more	40
3.9 Modeling result of the second version	43
3.10 Testing the second version	44
4 Programming a 5-axis system	47
4.1 Original system	47
4.2 5-axis system	47
4.3 Tool vector to angles	48
4.4 Transformation	49
4.5 Coding the 5-axis system	51
5 Printing with a 5-axis system	55
5.1 G-code	55
5.1.1 Creating G-code	56
5.2 Using the G-code	62
5.2.1 Calibrating the printer	63
5.2.2 Printing with the G-code	64
III Conclusion and Results	69
6 Results and Analysis	71
6.1 Modeling results	71
6.2 Printing results	72

7 Discussion and Conclusion	79
7.1 General discussion	79
7.1.1 The Pentarod	79
7.1.2 System compatibility	80
7.1.3 Test prints	80
7.1.4 Advantages with a 5-axis FDM system	80
7.2 Conclusion	80
7.3 Future work	81
A G-code generator Code	89

List of Figures

1.1	Side view of a FDM print	4
2.1	Illustration of the a FDM printer	8
2.2	Configuration of Delta 3D printer and Cartesian 3D printer . .	9
2.3	Multi axis systems designations and directions	10
2.4	Slicing a solid model	11
2.5	Illustration of Cutter Location coordinates {X, Y, Z} and its tool vector {I, J, K}	13
2.6	Illustration of backlash	13
2.7	The Duet and DueX4	16
2.8	A typical stepper motor	16
2.9	Example of a worm drive	17
3.1	The RepRap Ormerod	23
3.2	Intersection point of the rotary axes	25
3.3	Wiring for Duet and DueX4	26
3.4	The Ormerod nozzle mount	28
3.5	A axis and Z' limits for Ormerod	29
3.6	Illustration of how the relation between A-axis and Z' distance are found	29
3.7	The evolution of the platform	31
3.8	Print plate attachment on the first version	31
3.9	Motor and worm gear system on the C-axis mount, for the first version	31
3.10	Motor axle bearing	32
3.11	Axle to rotary system link, for the first version	32
3.12	Rotary system on the first version	32
3.13	First version nozzle mount	33
3.14	A axis and Z' limits for First version	34
3.15	The evolution of the X-axis	34
3.16	Top mount for Z-axis	35
3.17	The first version of the 5-axis FDM printer	35
3.18	Angular error A-axis, first version	36
3.19	Angular error C-axis, first version	37
3.20	Motor and worm gear system on C-axis for the second version	38
3.21	Sliding motor mount on the second version	39
3.22	Rotary system on the second version	39
3.23	The platform of the second version	40

3.24	The nozzle mount of the second version	41
3.25	The X-axis of the second version	41
3.26	A axis and Z' limits for the second version	42
3.27	The second version of the 5-axis FDM printer	43
3.28	Angular error A-axis of the second version	44
3.29	Angular error C-axis of the second version	45
4.1	Spherical polar coordinates visualization	49
4.2	Illustration of the transformation along A-axis	49
4.3	Illustration of the transformation along C-axis	50
4.4	Illustration of how the system handles G-code	52
4.5	Illustration of C-axis movement	53
5.1	Test structures	56
5.2	G-code simulation in HSMWorks	57
5.3	Visualization of spherical cap	58
5.4	The parts of the support test	59
5.5	G-code simulation of surface smoothing test on xy-plane	62
5.6	G-code simulation of support test	62
5.7	G-code simulation of surface smoothing test on xz-plane	63
5.8	Bad result of surface smoothing tests	65
5.9	Result of surface smoothing test on xy-plane	65
5.10	Result of surface smoothing test on xz-plane	66
5.11	Bad result of support test	67
5.12	Result of support test	67
6.1	Position of parts when using microscope	73
6.2	The print time for surface tests	74
6.3	Magnified view comparison for XY-surface and part with 0.1 mm layer height	74
6.4	Magnified view comparison for XY-surface and part with 0.1 mm layer height, top of parts	75
6.5	Magnified view comparison for XY-surface and part with 0.2 mm layer height	75
6.6	Magnified view comparison for XY-surface and part with 0.4 mm layer height	75
6.7	Magnified view comparison for XZ-surface and part with 0.1 mm layer height	76
6.8	Magnified view comparison for XZ-surface and part with 0.1 mm layer height, top of parts	76
6.9	Magnified view comparison for XZ-surface and part with 0.2 mm layer height	76
6.10	Magnified view comparison for XZ-surface and part with 0.4 mm layer height	77
6.11	Comparison of the part with support structure and the same part printed on the 5-axis system	78
6.12	The print time for support tests	78

List of Tables

3.1	Stepper motor specification	25
3.2	Motor accuracy in the worst case of the first version	27
3.3	Motor accuracy in the worst case for the second version	38
3.4	Results of A-axis stress test for the second version	45
3.5	Results of C-axis stress test for the second version	45
4.1	Motor steps per unit.	48

Glossary

CNC Computer Numerical Control

CAD Computer-Aided Design

CAM Computer-Aided Manufacturing

FDM Fused Deposition Modeling

FFF Fused Filament Fabrication

SLA Stereolithography

SLS Selective Laser Sintering

LDM Laser Metal Deposition

PLA Polylactic acid

ABS Acrylonitrile Butadiene Styrene

DIY Do It Yourself

G-code A language for describing tool paths for CNC and 3D printers

CL Cutter Location

MCS Machine Coordinate System

ISO International Organization for Standardization.

ROBIN Robotics and Intelligent systems research group at the University of Oslo.

Part I

Introduction

Chapter 1

Introduction

1.1 Introduction

The earliest 3D printing technology surfaced already in the late 1980's. However, until the late 2000's 3D printing technology was still expensive and mostly used by industry. In 2004 Adrian Bowyer conceived the RepRap concept of an open source and self-replicating 3D printer. During the next years Bowyer and his team at the University of Bath developed working prototypes of 3D printers, based on this concept. In January 2009 the first commercially 3D printer was offered for sale, in kit form and based on the RepRap concept. This printer used the Fused Filament Fabrication (FFF) more commonly known as Fused Deposition Modeling (FDM). FDM printers builds structures layer by layer from the bottom-up, by heating and extruding material on a build plate.

This marked a turning point for the 3D printer technology, after which 3D printers became commercially available. From this point onwards 3D printing technology became extremely popular. Both cheap and do it yourself (DIY) printers have surfaced using different 3D printing technologies. Today the FDM printers are the most popular 3D printers. This is most likely because the FDM printers are consumer friendly, can create rigid parts, and are low cost [50].

1.2 Problems with FDM printers

FDM printers still have some weaknesses including accuracy and their inability to print overhanging structures without support [50].

The accuracy can be improved to some extent by lowering the layer height and using a nozzle with a smaller diameter. Even though the accuracy is improved by these methods, there will still be problems when printing surfaces that are not parallel with the print plate. An example of this can be seen in Figure 1.1.

When improving the accuracy of the printer by lowering the layer height or using a nozzle with a smaller diameter, the print time for a part will be extended. As an example, by lowering the layer height from 0.4 mm to 0.2 mm, the amount of layers will be doubled and accordingly the print time

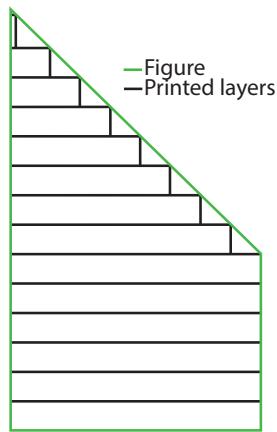


Figure 1.1: Side view of a FDM print

will approximately double. When using a nozzle with a smaller diameter the print time will also increase, because the volume of the plastic the printer can extrude will be dramatically reduced [58]. Time is also a problem when using support material to print structures with overhangs, as the support structure creates more work for the FDM printer.

1.3 Possible solutions

FDM printers build up parts with the additive manufacturing process, the opposite process is called subtractive manufacturing. A common technology that uses the subtractive manufacturing process are CNC mills. CNC mills are controlled in a similar manner as FDM printers, but remove material from a work piece instead of adding it.

For CNC mills the surface problem have been solved in three ways: different types of cutting tools, tool paths that are coordinating movements in three dimensions instead of only two and multi-axis mills. Tool paths movements in three dimensions would not solve the support structure issue of FDM printers, and different cutting tools are not an option in a FDM printer. In multi-axis mills, the surface problem has been solved by rotating the work piece or the tool, so that the tool can mill from different angles, and thereby create a smooth surface. The most common multi-axis configuration when working with different structures, is a 5-axis mills.

The weakness a FDM printers suffer today is that the printed parts are always printed bottom-up. By rotating either the nozzle or the printed part, as done in 5-axis mills, these problems could be eliminated. Therefore, this thesis will focus on the creation of a 5-axis FDM printer.

1.4 Goals of the thesis

As mentioned in section 1.2 FDM printers have two main weaknesses; accuracy and the ability to print overhanging structure without support.

Both of these faults can be addressed to some degree on a 3-axis FDM printer, but they are time consuming, and even when the accuracy is increased the final part still have flaws on surfaces not parallel with the print plate.

The objective of this thesis is to see if a 5-axis FDM printer can solve these issues. To test this theory, an already developed open source FDM printer will be extended with two rotary axes, to work as a 5-axis FDM printer. Then the two weaknesses, accuracy and ability to print overhanging structure without support, of a FDM printer will be tested to see if a 5-axis FDM printer can eliminate these without severely extending the print time.

Since this system will be based on an open source printer, the 5-axis system will be released as an open source project when this thesis is finished.

1.5 Outline of the thesis

This thesis is divided into 3 parts; Introduction, The Project, and Conclusion and Results. The three parts are again branched into 7 chapters: Introduction, Background, Building a 5-axis system, Programming a 5-axis system, Printing with a 5-axis system, Results and Analysis, and Discussion:

- The “Background” chapter explains the relevant information for this thesis, and then some of the previous work and research relevant for this project is provided. This chapter also lists the tools and programs used to realize this thesis.
- The “Building a 5-axis system” chapter explains the designing and building procedures for the system. This chapter is divided into four parts. The first part explains the choice of design and the requirements of the system. The second part point out the planning and building process of the first version. The third part discuss the results and problems with the first version. The last part explains the planning and building process of the second and final version.
- The “Programming a 5-axis system” chapter explains how the original firmware is working and how it is modified to work as a 5-axis system.
- The “Printing with a 5-axis system” chapter starts with explaining how the G-code for testing the system was created, then the printing process with the printer is explained.
- The “Result and analysis” chapter present and analyze the final version and its printing results.
- The last chapter starts by discussing the results, followed by a conclusion and suggestions for future work.

Chapter 2

Background

The background theory is presented in this chapter. Also, some discussion regarding central aspects of this thesis is made.

2.1 Computer Numerical Control (CNC)

When movement and tool handling is controlled by a computer it is typically a computer numerically control system (CNC). A CNC system is normally built up with 3-axis, X and Y for movement of the work piece and Z for movement of the tool. With a CNC system it is possible to get a high precision along the axis, with the help of stepper motors or servo motors. These motors are usually geared down to get an even better precision. In a typical CNC system the operator creates a series of steps from a computer aided design (CAD) and the machine converts these steps into movements to create a part that closely matches the original CAD.

2.2 Additive Manufacturing

Additive manufacturing is a process where digital 3D design data is used to build up a 3D structure layer by layer, by depositing material. The material used by additive manufacturing can be plastic, liquid, powder filaments or even paper [4, 57]. The chosen material is dependent on which additive manufacturing method that is being used. Some of the different additive manufacturing methods that exists today are:

- Stereolithography (SLA)**

SLA is the oldest additive manufacturing method. It was patented in 1987 by Charles Hull, co-founder of 3D Systems Inc. SLA uses liquid material, which is cured and solidified by an ultraviolet laser light. The laser light traces a cross section of the part which is being created. When one layer is done the part is lowered by a distance equal to a specified layer height, and a resin filled blade sweeps over the part, re-coating it with new material. This is repeated until the part is finished [34, 57].

- **Fused deposition modeling (FDM)**

FDM was developed and implemented the first time by Scott Crump in 1980s and was commercialized in 1990 by Stratasys. Other 3D companies have adopted similar technologies but under other names, such as Fused Filament Fabrication (FFF). FDM printers builds structures layer by layer from the bottom-up by heating and extruding thermoplastic filament such as Acrylonitrile Butadiene Styrene (ABS) and Polylactic acid (PLA). The extruder travels between points in the X, Y and Z coordinates. These points are found by slicing a CAD model into layers defined by the layer height, and computing traveling paths. To support overhanging layers, support material is often used in FDM printer. The support material can be dissolved after a print is complete [34, 57]. An illustration of a FDM printer can be seen in Figure 2.1.

- **Selective Laser Sintering (SLS)**

SLS was developed by Carl Deckard, a student of Texas University, and his professor Joe Beaman in 1980s. The SLS technology is in some ways quite similar to SLA. The main difference between the two is that SLS uses powdered material instead of liquid. Unlike the methods described above, SLS systems do not need support structure since the object being printed is constantly surrounded by unsintered powder [57].

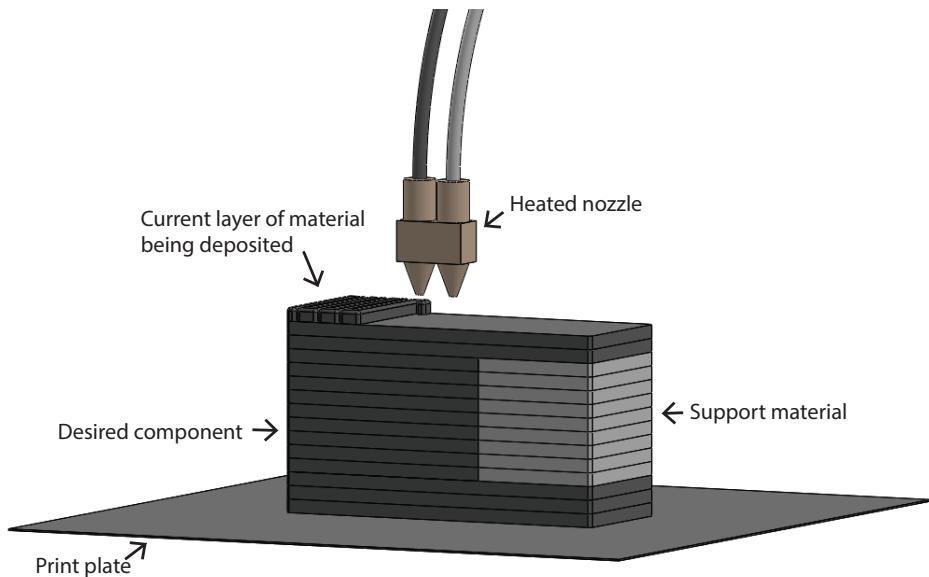


Figure 2.1: Illustration of a FDM printer.

2.2.1 FDM printers

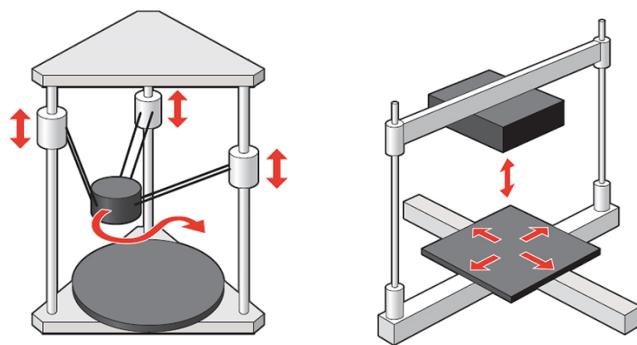
As mentioned in section 1.1 FDM printers are the most popular 3D printers today. Because of their popularity, simplicity and availability in the open source community a FDM printer was chosen for this thesis. FDM printers exists in multiple different configurations, the two most common types being:

- **Cartesian**

The Cartesian 3D printers consist of three rails that have movement along the axes of the Cartesian plane. Controlling the movement of a Cartesian 3D printer is fairly simple mechanically and software wise as there is only linear movement along the axis. Most of the 3D printers on the market today use the Cartesian system [10]. An illustration of the configuration and movement of a Cartesian 3D printer can be seen in Figure 2.2b.

- **Delta**

The Delta 3D printers consists normally of three vertically rails that are standing upright in a triangular configuration. The extruder is connected to these rails with three arms, and its movement is achieved by moving them up and down. Due to this, it is more mathematically complex to find the head position of a Delta printer, which makes the Delta printers more advanced software wise. Delta printers are becoming more popular in both industry and the 3D printing consumer community. This is because the Delta printers configuration enables faster printing and a more compact size than the Cartesian printers [10]. An illustration of the configuration and movement of a Delta 3D printer can be seen in Figure 2.2a.



(a) Delta 3D printer. Printer head can move in any direction quickly

(b) Cartesian 3D printer. Each element moves only in one direction.

Figure 2.2: Illustration of Delta 3D printer (a) and Cartesian 3D printer (b) [26].

Both the Delta and the Cartesian configuration would work for this thesis, but because of availability the Cartesian type were chosen for this project.

2.3 Subtractive Manufacturing

Subtractive manufacturing is a collection of various controlled processes where a piece of raw material is being removed to create a desired final shape. Some of these processes are drilling, turning, milling, grinding and chip formation. In the recent years it has become more common to entrust the control of the process to CNC systems. [2, 31].

The CNC milling machine typically operates the same way as a Cartesian 3D printer, but it removes material instead of adding it. Because of this similarity between 3D printers and CNC milling machines, and the fact that multi-axis CNC mills already have existed a while. Multi-axis CNC mill have been the main inspiration for the 5-axis 3D printer.

2.3.1 Multi-axis CNC mill

When a CNC system has more than 3-axis it is called a multi-axis system. Now the computer also controls rotation (ABC) and parallel movement (UVW) along X, Y or Z, as shown in Figure 2.3. (ABC) specifies a rotation around an axis, and (UVW) specifies a parallel movement along the axes. For example, it is possible to move the work piece along the X-axis in addition to moving the tool along the X-axis.

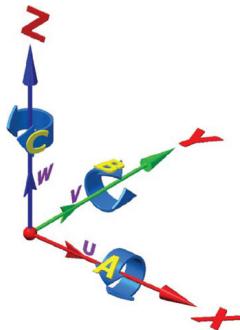


Figure 2.3: Multi axis systems designations and directions [6].

The 5-axis CNC machine is one of the most used multi axis systems for creating different parts when high precision is needed [62]. The two extra axes in a 5-axis system tend to be rotation axes. These rotations can either be done by rotating the head, where the tool is attached, or rotating the table, where the work piece is attached. Therefore, the 5-axis system can have three different configurations:

- **Table/Table**

Table/Table machines are the most common types of multi-axis machines. In a table/table system all the rotation, except the spindle, is done with the work piece. This works with tables that are rotating around the X-, Y-, and Z-axes. This way the part is physically rotated around the tool, therefore the machine's rotary devices need to be capable of handling the weight of the part and the fixture [6].

- **Head/Head**

In a Head/Head machine all the rotary motions are executed by the spindle head of the machine. These machines can be both vertical and horizontal. When both of the rotary axes are in the head it is possible to change the whole head. This makes it possible to achieve different behaviors based on the configuration of the head [6].

- **Head/Table**

Head/Table machines are a combination of the two previous configurations. Head/Table machines have one rotary axis where the tool is attached, and one at the table of the machine. For milling Head/Table machines are arguably the most capable of these three groups, and they can machine large, heavy parts [6].

2.4 Tool path generation

To be able to create a structure with a milling machine or 3D printer, the tool path from a 3D model has to be generated. This is done by slicing the triangular polygon, from a geometry file, into horizontal layers [34]. The more layers the structure is sliced into the more accurate the geometry approximation will be. However the printing or machining process of the structure will take a longer time. Each polygon is converted to a contour line between the polygons and the slicing plane to find the outline of the model. The process from the solid model to a sliced model can be seen in Figure 2.4. When all the contour lines are found, they are sorted and the interior/exterior area of each contour is found. Tool paths filling in the interior for 3D printing or exterior for CNC machining, are then generated and written to a file as for example standard G-code [34].

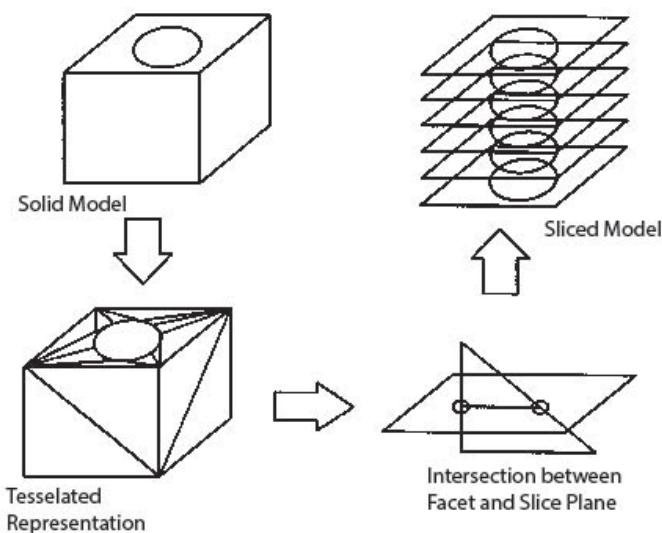


Figure 2.4: Illustration of how a solid model is being sliced [56].

2.5 G-code

G-code was originally defined by the Electronics Industry Association (EIA) in the 1960's [25]. G-code is the common name for the most widely used numerical control programming language [31]. The G-code specifies what a machine should do using instruction lines. It can be used to change the settings of the machine, put it in special modes or tell it where it should move its tool. The simplest form of G-code can be used to specify linear interpolation. This is done by sending the G-code command $G01/G1$ with X, Y, Z, F, where X, Y and Z is the position the tool should move to in the Cartesian coordinate system, given as mm or inch. The feedrate (F) defines the movement speed of the tool tip, normally interpreted as mm/min or inch/min. To keep the file size of the G-code to a minimum, the G-code is modal. This way if the G-code does not get a new value for an axis, is kept in the old position [34].

2.5.1 5-axis G-code

5-axis G-code has all the functionality as the ordinary G-code, but additionally commands the rotation of the two extra axes. The most common way to command the rotation is by adding A, B and C to the G-code. A, B and C define the rotation angles, an illustration of this can be seen in Figure 2.3. When A, B and C are used to represent the rotation in a system, the G-code will be dependent on the machine configuration the G-code is created for. To avoid this issue it is possible to represent the rotation by using Cutter Location (CL) points represented in ISO format as $\{X, Y, Z, I, J, K\}$, where $\{X, Y, Z\}$ are the coordinates of CL and $\{I, J, K\}$ are the tool vector values [15, 43]. When representing the rotation as $\{I, J, K\}$ the machine can compute which axes it should move to get to that position, based on its configuration. An illustration of the tool vector, and its values in a Cartesian coordinate system, can be seen in Figure 2.5.

Tool vectors are also a great way to simplify post processor creation. When a 5-axis tool path is programmed using a CAM system, the software already computes the commanded moves with tool vectors. Therefore, by using tool vectors in the machine, the post processor can simply output these created moves without having to translate them. When using $\{A, B, C\}$ representation the post processor have to translate every move into an A-, B- or C-axis [15].

To create a system which can use G-code that is not machine depended, and to have a G-code which are easier to comprehend, the machine created in this thesis uses 5-axis G-code with the ISO format ($G1 \{X, Y, Z, I, J, K\}$).

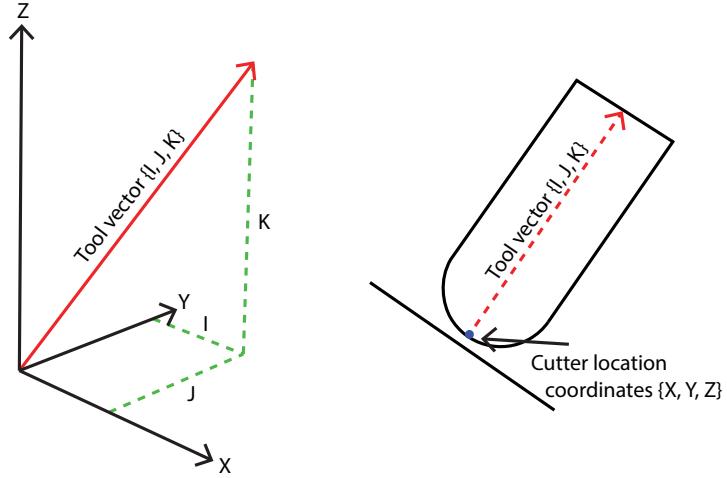


Figure 2.5: Illustration of Cutter Location coordinates $\{X, Y, Z\}$ and its tool vector with the values $\{I, J, K\}$

2.6 Backlash

Backlash is the maximum distance or angle two parts in a mechanical system can move without applying a force or motion to the next part in the mechanical sequence [8]. In gears this distance is measured as the clearance between two mated gear teeth. This clearance is necessary to avoid e.g. interference, wear, and excessive heat generation. It also ensures proper lubrication and compensates for manufacturing tolerances. [29, 33]. In Figure 2.6 the concept of backlash is illustrated.

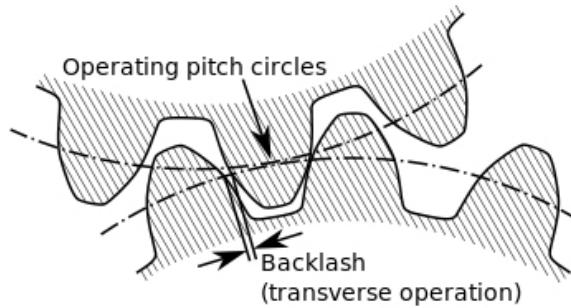


Figure 2.6: Illustration of backlash [11]

One common method to minimize or eliminate backlash is preloading. Preloading means to apply a constant force so the clearance between two parts, or two teeth for gears, is always minimized [33].

2.7 Open source

Open source is a develop model where the developer distributes their code and machine plans freely. End users are encouraged to modify both the plans and the code, and then share their modifications. This way the creators can get new ideas and inspiration on how they can improve their machines. In the open source community the profit of a product is not the core incentive, but instead they are driven by curiosity and the desire of solving new problems [40, 41].

The plan with this thesis is for it to be open source, and it will be published on a suitable site online when the master's thesis is completed.

2.8 The RepRap project

The RepRap project started with an idea Adrian Bowyer published in February 2004. The idea was to make an affordable 3D printer which could print out most of its own parts, and could be assembled with parts that were cheap and easy to get. RepRap follows the principles of the Free Software Movement, and therefore distributes all the RepRap machines as open source [3]. The first four official 3D printing machines made by the RepRap project where released in March 2007 named "Darwin". After this, both the RepRap core team and the community have made multiple 3D printer models and variations of these.

2.9 Previous work

When work on this thesis began there was only one other system that where capable of 5-axis 3D printing, but during the time spent on this work more 5-axis 3D printers have surfaced. Some of these are:

- **DMG Mori:**

January 10, 2014 DMG Mori released a video¹ of a hybrid machine capable of both subtractive and additive manufacturing of metal. This machine has a table/table configuration and is able to change its tool depending on which kind of machining it is doing. Other than an article posted 24. October 2014² the information about this machine has been sparse.

- **TWI:**

On November 14, 2014 TWI released a video³ of a 5-axis Laser metal deposition (LDM) printer for metal. This printer also uses the table/table configuration and uses LDM technology, developed by TWI, to print the metal. The information about this printer has been sparse except for the general information TWI has released.

¹DMG Mori 5-axis hybrid: <https://www.youtube.com/watch?v=s9IdZ2pl5dA>

²Article DMG Mori: <http://www.engineering.com/AdvancedManufacturing/ArticleID/8778/3D-Printing-and-5-Axis-Machining-Combined-in-One-Machine.aspx>

³TWI 5-axis LDM machine: <https://www.youtube.com/watch?v=yKnlmfuMSgo>

- **5axismaker:**

September 22, 2014 5axismaker created a Kickstarter for a 5-axis multi-fabricator. The idea behind this machine is to create a 5-axis system in which it is possible to change tools on. The set of tools would be a CNC mill, touch-probe, 3D printer, wire-cutter and others [1, 39]. When this machine was put on Kickstarter, the 3D printer head was still in development. On June 10, 2015 they released a video⁴ of the 3D printer capability of their machine. This machine has a head/head configuration.

5-axis 3D printers do exist today, but either the information about them is difficult to find or they are still in development. Because of this the main inspiration of this thesis comes from 5-axis CNC milling machines, since it is a more developed field and it is better documented. Although the 5-axis CNC field is more developed and better documented, were no public available information about how the rotary system of 5-axis CNC machine are built found during the research of this thesis. Therefore will the 5-axis rotary system in this thesis be build from scratch.

2.10 Tools and programs used

This section describes tools and programs used in this thesis.

2.10.1 Duet

The Duet was developed by Andy Hingston and Tony Lock from Think3dPrint3d in conjunction with RepRapPro. The Duet combines the Arduino Due micro controller with four stepper motor controllers, Ether-net, HI-Speed SD card slot and more [12, 42].

The Duet is the 3D Printer controller board used in this thesis.

Duemilanove

The Duemilanove is an expansion card for the Duet. It has four extra stepper drivers, another I2C digipot, 4 FETs and corresponding thermistor inputs [42]. The Duemilanove was developed to make it easy to add multiple extruders to a 3D printer. The Duet and Duemilanove can be seen in Figure 2.7.

In this thesis, the Duemilanove is used to control the extra motors the 5-axis system needs.

⁴5axismaker printer: <https://www.youtube.com/watch?v=w8Fl8L4yk8M>

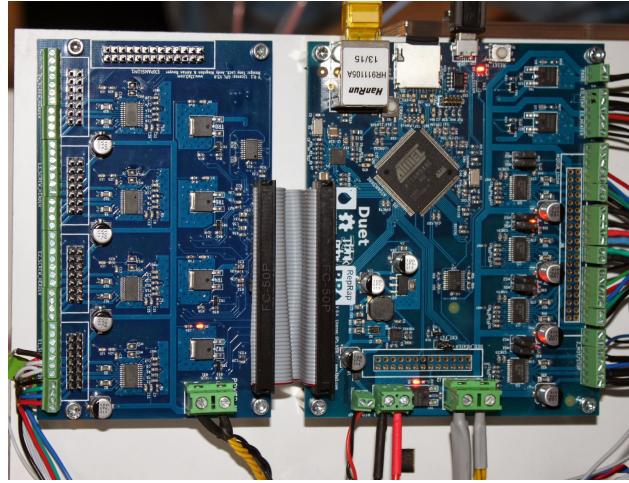


Figure 2.7: The Duet at the bottom and the Duex4 at the top [42]

2.10.2 Stepper motors

Stepper motors are used in this thesis to control all the axes and the extruder.

A stepper motor is driven by a magnetic gear with multiple teeth that are connected to the rotor in the motor. In the stator there are electro magnets which are equipped with corresponding teeth, and the rotor rotates when these electro magnets are turned on and off in a specified sequence. Since the magnetic force is heavily dependent on the air gap between the magnets, a small step size can be achieved. Normally a stepper motor has a step size between 200-400 steps each revolution [32]. A stepper motor has three ways to get input for the movement; full step, half step and micro step. When using full step, the drives always have two phases on. In half step the drives alternates between two phases on and single phase on. Half step has a double resolution over full step, but the average torque is lower. To get micro step the driver sends an AC waveform to the motor, which usually is sinusoidal. This way the phases pulls and pushes on the teeth of the iron gear with different strengths, and thereby we get multiple steps where full step only gives us one [32].



Figure 2.8: A typical stepper motor

2.10.3 Worm drive

Worm drive was used in this thesis to gear down and transfer the movement of the rotary axes.

A worm drive is a gear arrangement, which consists of a worm and a worm gear. The worm has the form of screw, while the worm gear has a shape that are similar to a spur gear. A Worm drive offers a high gear reduction, but still has a small size [49, 63]. As seen in Figure 2.9 the worm, and the worm gear has its drive axes at 90° to each other.

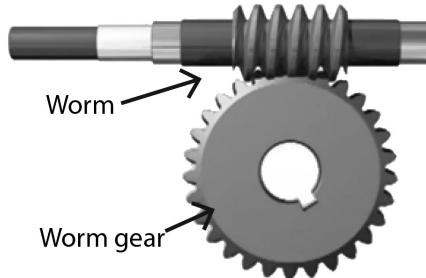


Figure 2.9: Example of a worm drive [49]

2.10.4 Fortus 250mc

In this thesis, the Fortus 250mc 3D printer is used to print most of the parts of the 5-axis system.

Fortus 250mc is manufactured by Stratasys and utilizes the Stratasys patented Fused Deposition Modeling (FDM) technology. Fortus 250mc uses ABSplus-P430 as filament, has a build envelope of 254 mm * 254 mm (base) * 305 mm (height), and can print with a layer height of 0.178 mm, 0.254 mm or 0.330 mm. To avoid wrapping of the ABS plastic the Fortus 250mc uses a heated camber that creates a thermally isotropic build envelope. The Fortus 250mc allows for a certain degree of overhang, but needs to use support structure when the overhang is over $30^\circ - 45^\circ$. The support structure can then be removed by breaking it, or by dissolving it in a chemical bath with high pH value (basic). Removing the support in a chemical bath usually takes some time, but it can remove support structures that are hard to reach [27, 28]. For generating tool paths for the Fortus 250mc the program Insight is being used. Insight prepares a STL file for 3D printing by optimizing build orientation, slicing, creating support structure and generating the tool path [38].

2.10.5 Objet Connex 500

In this thesis the Objet Connex 500 3D printer is used to print parts where a high accuracy was needed.

The Objet Connex 500 was created by Objet Geometries, which merged with Stratasys in 2011. The Objet Connex 500 uses the polyjet technology, where the model is built by applying a layer with thin liquid photopolymer

which is instantly cured with the use of UV lamps. This is done for each layer until the model is finished [44, 45]. The Objet Connex 500 can print with multiple materials on the same print, and can mix these material to create materials with different properties. The printer has a build resolution of 600 dpi for the X-axis, 600 dpi for the Y-axis and 1600 dpi for the Z-axis. The printer can print with a layer height down to 16 micrometer [44, 46].

2.10.6 Epilog Zing 24 Laser 30W

The Epilog Zing 24 Laser used for this thesis have a laser wattage of 30W.

Epilog Zing 24 Laser is manufactured by Epilog Laser. It is one of their Zing Laser Series. The Epilog Zing 24 has a work area of 610 mm * 305 mm and can cut in multiple materials as long the material does not create poisonous gases when heated up.

2.10.7 SolidWorks

In this thesis, SolidWorks have been used to design and test the different versions of the 5-axis 3D printer. There exists multiple CAD software producers [13], but SolidWorks was chosen because the University of Oslo has licenses for it.

SolidWorks is perhaps the most popular CAD software available today. When building a model in SolidWork the designer starts with sketching a 2D model, which is extruded into a 3D model [52].

2.10.8 HSMWorks

HSMWorks is a CAM extension for SolidWorks. HSMWorks is made to create a seamless workflow between a CAD and CAM software. HSMWorks has the ability to create tool path strategies for 2D, 3D and multi-axis milling. HSMWorks also includes machine simulation, which can simulate tool paths on a machine assembled in SolidWorks and a backplot tool which can simulate G-code [35].

HSMWorks was used in this thesis to generate multi-axis tool paths, simulate these tool paths on the 5-axis 3D printer with the machine simulation and simulate manually generated 5-axis G-code with the backplot function.

2.10.9 Printrun

Printrun was used in this thesis to control the 5-axis 3D printer and send G-code to it, with prонterface.

Printrun is a set of G-code sending applications that are free and open source. It consists of: a command line G-code sender called prонsole, a dumb G-code sender called printcore, a G-code sender with graphical user interface called prонterface and other helpful scripts. Printrun was created by Kliment Yanev [47].

2.10.10 G-code generators

A G-code generator is a program that is cutting a 3D model into a number of slices of a given height, and describing each slice as the path the printer head should follow. This path is described as multiple points in a coordinate system, where each point is a G-code command. In this thesis two G-code generators were used:

- **Cura**

Cura is a free open source software developed by David Braam. Cura has a graphical user interface which is easy to use, has the option to add add-ons, and it is possible to adjust the settings of the slicer multiple ways [20].

- **Slic3r**

Slic3r is a free open source software that was born within the RepRap community in 2011. Slic3r was created to provide the 3D printing technology with an open and flexible tool chain. Slic3r has features as multiple extruders, brim, micro layering, bridge detection, command line slicing, variable layer heights, sequential printing (one object at time), honeycomb infill, mesh cutting, object splitting into parts, AMF support, avoid crossing perimeters, distinct extrusion widths, and much more [51].

Slic3r and Cura were used in this thesis to create G-code for the base and top structure of the 5-axis test parts.

2.10.11 C++

C++ is the main used programming language used for the Ormerod firmware, therefore the control of the 5-axis system is written in C++.

C++ is a general-purpose programming language with imperative, object-oriented and generic programming features. It also provides possibilities for low-level memory manipulation. C++ was developed as “C with classes” in 1979 by Bjarne Stroustrup. Later in 1983, it was renamed to C++. In 1988 C++ was standardized by International Organization for Standardization (ISO) [55].

2.10.12 Java

Java was used in this thesis to manually generate 5-axis G-code, Java was chosen because of its familiarity.

Java is an object oriented computer programming language that is concurrent and class based. Java was released in 1995 by Sun Microsystems [61]. When they were creating Java they wanted to make a language that was simple, object oriented, sturdy and secure, platform independent, and still had a high performance [24].

Part II

The Project

Chapter 3

Building a 5-axis system

When building the 5-axis system, two versions were created. This chapter will explain the planning, building, and testing process of both of these versions. The problems with the first version and the reason to create a second version will also be explained in this chapter.

3.1 Planning the 5-axis system

3.1.1 Choosing a base system

When looking for a 3-axis 3D printer that could be expanded to work as a 5-axis system, the 3D printers from the RepRap community was an obvious choice. Since they are designed to be easy to build and modify. The RepRap community has multiple 3D printers that would be suitable for this project. In the beginning of this thesis ROBIN's main supplier of components (RS-components), started to sell the RepRapPro Ormerod. Therefore the Ormerod was chosen for this project. This printer is also a good choice since it is easy to expand upon without too many significant changes to the design.

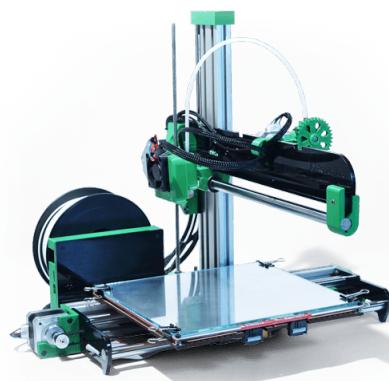


Figure 3.1: The RepRap Ormerod [48]

3.1.2 Choosing 5-axis configuration

Working 5-axis mills already exist. These have been the main inspiration for this 5-axis printing system. 5-axis systems consists of three main configurations which are; table/table, head/head and table/head, as were explained in section 2.3.1. In this project it was decided to use a table/table configuration, the main reasons for this are:

- Two extra axes would have extra weight. Because of this extra motors have to be added. In a head/head or head/table system these motors have to be added to the X- and Z-axis of the Ormerod. This could prove more difficult than adding extra motors on the Y-axis, which a table/table system needs.
- A head/head system or a head/table system needs to be compact at the head to work properly. This will make the system harder to assemble and redesign. Whereas a table/table system can easily be built larger as long the weight is compensated with more and/or stronger motors.
- As mentioned in section 2.3.1, one of the benefits of a head/head or head/table system over a table/table system, is that it can work with heavier work pieces. This system will print with plastic, and therefore the weight will not be an issue.
- One of the problems a table/table have is the attachment of the work piece. This could be an issue with printing on a table/table system, since adding attachment to a part which is being printed could be difficult. After testing a few prints on the Ormerod, this was found not to be an issue. Since the plastic sticks quite well on the print plate, and force must to be applied to remove it.
- Another problem that might come with a head/head and head/table configuration is that the plastic might not connect completely or start floating down the structure when printing 90° on an existing structure. While there are no research that can support this claim, a table/table system is preferable to be sure this will not be an issue.

3.1.3 Finding the origin in the system

In a table/table configuration the two rotary axes will often intersect, therefore the most convenient location of the machine coordinate system (MCS) origin will be were the rotary axes intersect [30]. An Illustration of the intersection point in a table/table system can be seen in Figure 3.2 where the MCS origin is in the intersection of the red lines. The reason to have the MCS where the rotary axes intersect is to ease both, the programming of the firmware and to have a G-code that is more comprehensible.

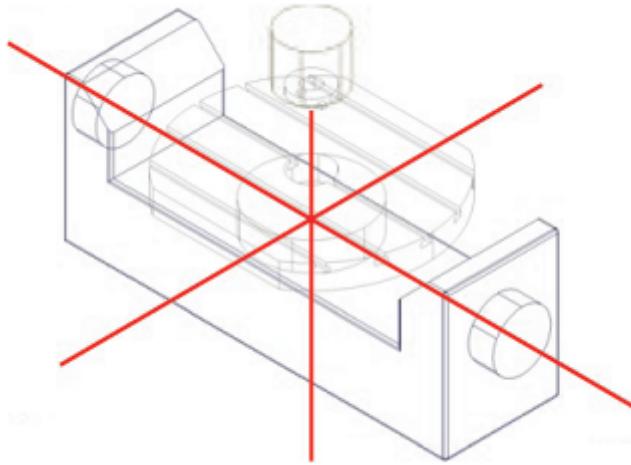


Figure 3.2: Intersection point of the rotary axes for a table/table configuration[14]

3.1.4 Choosing stepper motors

The motors that were included in the Ormerod kit are the JK42HS34-1334A shown in Section 3.1.4. For the 5-axis system three more motors were needed: one extra for the Y-axis because of added weight, one for the A-axis and one for the C-axis. Because of availability, the SM-42BYG011-25 shown in Section 3.1.4 were chosen as the extra motors. As seen in Section 3.1.4, these two motors are similar in most cases, and therefore they can be added to the system without any modifications.

Model No.	JK42HS34-1334A	SM-42BYG011-25	42BYGHM809
Step Angle	1.8°	1.8°	0.9°
Step angle accuracy	±5%	±5%	±5%
Step per revolution	200	200	400
Current/Phase	1.33A	0.33A	1.7A
Holding Torque	2.2 kg*cm	2.3 kg*cm	4.9 kg*cm
Detent Torque	120 g*cm	163 g*cm	224 g*cm
Size (WxLxH)	42x42x34 mm	42x42x34 mm	42x42x48 mm

Table 3.1: Stepper motor specification [21, 22, 37]

3.1.5 Controlling the stepper motors

RepRap Ormerod is using an open source 3D printer controller board called Duet. Duet combines the Arduino Due micro controller with four A4892 stepper drivers, Ethernet, Hi-Speed SD card slot, High current PWM, 12-35v DC power input and other components [12]. The A4892 stepper drivers have the possibility to go to 1/16 micro step, which provides high accuracy. As seen in Figure 3.3 the 5-axis system requires a total of seven motors and therefore seven motor drivers. In addition to the Duet controller, the

RepRap community has also built an expansion board for the Duet called Duex4. The expansion board is built to create the possibility to add four more extruders to the 3D printer. Since the Duex4 uses the same stepper drivers as the Duet it is possible to use these four extra stepper drivers for the extra axes, as shown in Figure 3.3.

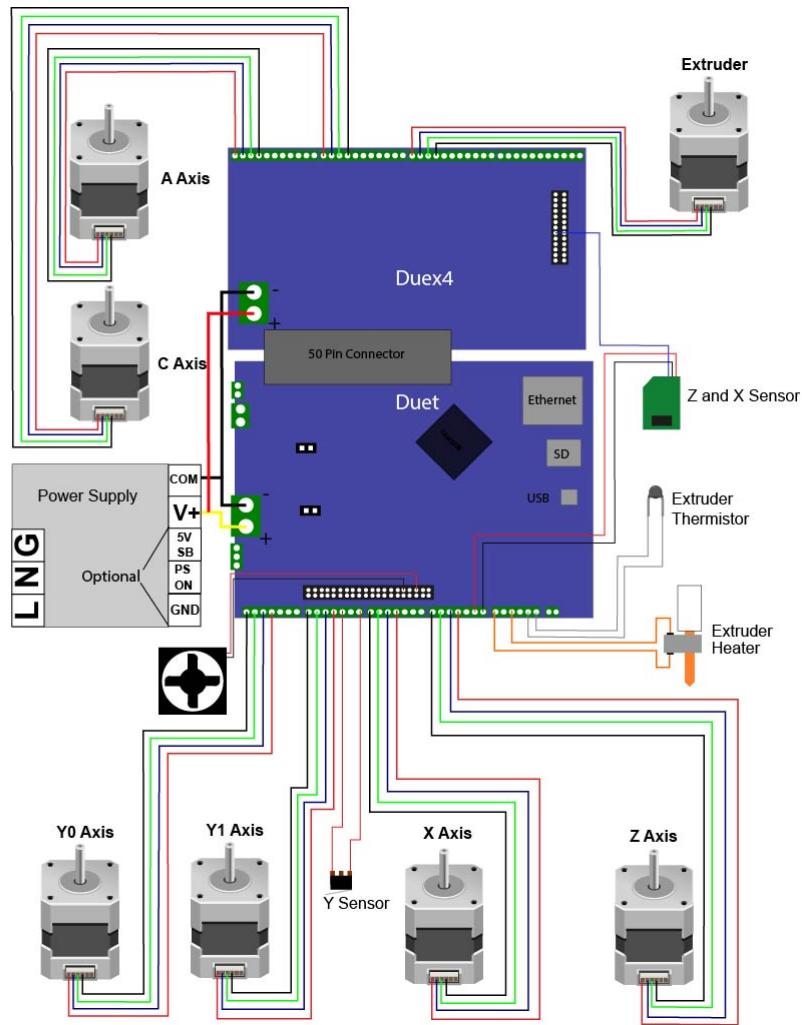


Figure 3.3: Wiring for Duet and Duex4. Figure based on Duet illustration [12]

3.1.6 Accuracy for the rotary system

In a 3D printer, a high accuracy is important for creating quality parts. The accuracy for the A- and C-axis can be shown in two ways, how many degrees it moves in each step or how large the arc length movement is at the end of the printer surface. To find the accuracy in the worst-case scenario, the best solution is to compute the arc length movement for each step. In the worst-case scenario it is possible to print on the outer edge of the printer surface which gives a radius of 70 mm. The step angle of the stepper motors

without micro steps is 1.8° , with $1/16$ micro step, the step angle for each step is 0.1125° . The arc length can be found using eq. (3.1) [9], where s is the arc length, r is the radius of the circle and θ is the angle in degrees.

$$s = \frac{\pi * r * \theta}{180} \quad (3.1)$$

The arc length movement for each micro step is found to be 0.13745 mm, which is 12 times higher compared to the accuracy of the X- and Y-axes, as seen in Section 3.1.6. To achieve the same accuracy a gear with ratio $1/12$ and higher is needed. When working with a gear ratio this high and with the need of a compact solution for the C-axis, a worm gear system is the best choice. This due to the system's compactness and high gear ratio, as mentioned in section 2.10.3. To make sure the A- and C- axes have a higher accuracy than the X- and Y-axes, a worm gear system with a $1/15$ ratio is being used for this system. The accuracy obtained with this is shown in Section 3.1.6.

Axis	$\frac{step}{mm}$	$\frac{mm}{step}$
X	87.489	0.01147
Y	87.489	0.01147
Z	4000	0.00025
A	109.17	0.00916
C	109.17	0.00916

Table 3.2: Motor accuracy in the worst case of the first version

3.2 Planning the first version

When the Ormerod was assembled and tested, it appeared there were a few modifications that needed to be done on the original design to make it support a table/table configuration. The table/table configuration would add a rotary system on top of the Y-axis. This gives the Y-axis motor more weight to move compared to the original print plate on the Ormerod. To compensate this an extra motor had to be added to the Y-axis to make sure the Y-axis would be able to move the rotary system. The rotary system would also raise the print plate and be wider than the original print bed. To compensate these changes the platform and the Z-axis had to be expanded.

As mentioned in section 3.1.3 it is convenient to keep the origin of the machine coordinate system (MCS) at the point where the rotary axes intersect. To make sure this is the case the design of the rotary system should be designed with the print plate in focus. The print plate should also have an adjustment ability to make sure the print plate is laying directly in this origin.

In the worm gear system that was chosen for this system the worm have a different hole diameter then the axle of the motor, 6 mm and 5 mm respectively. One solution to this problem would be to add a lining between the worm and the axle of the motor.

When the A-axis is rotated to an angle, the nozzle must be able to reach the print plate. In the design of the Ormerod the nozzle mounting has obstructions on both sides, as can be seen in Figure 3.4. One of these sides is the cooling system for the nozzle and needs to be there, while the other side can be improved upon. To avoid that the system has different relations between the print plate and the nozzle is depending on the A-axis being negative or positive, whereas the A-axis will be limited to $0^\circ - 90^\circ$. By doing this the gear system in the A-axis will also be preloaded by the weight of the C-axis mount. This will, as mentioned in section 2.6, eliminate problems with backlash in the A-axis. The relation between the A-axis angle and the Z' distance between the nozzle and the print plate (for the original nozzle mounting) can be seen in Figure 3.5. The values in this graph has been computed by measuring the distances in SolidWorks and then computing the A-axis angle and Z' distance. An illustration of how these values were found can be seen in Figure 3.6.

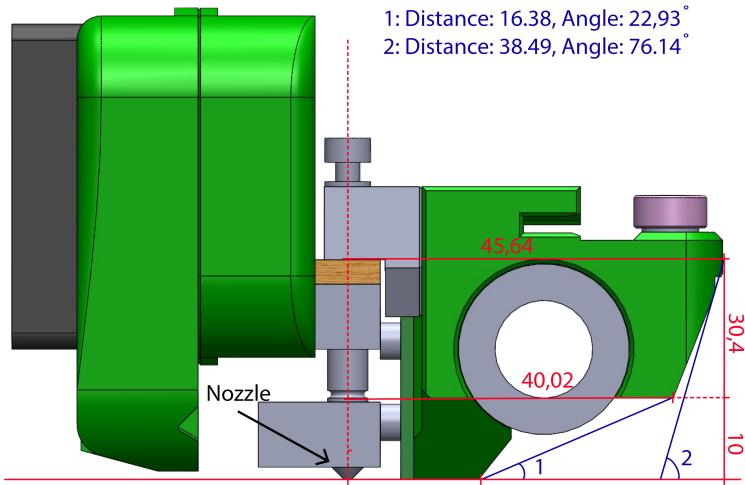


Figure 3.4: The Ormerod nozzle mount, all distances are in mm.

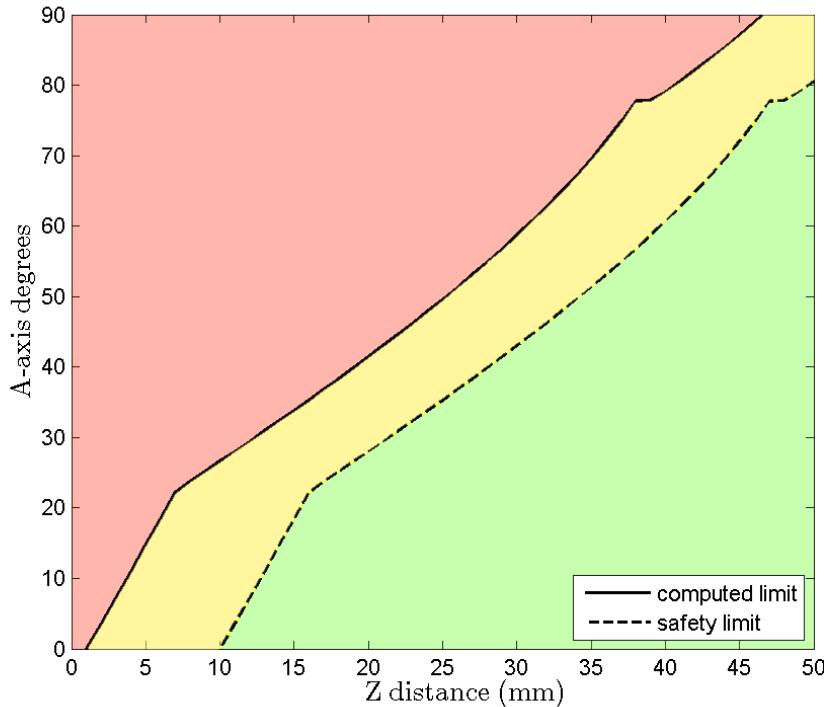


Figure 3.5: A-axis and Z' limits for Ormerod. Computed limit is found as mentioned in section 3.2 and safety limit is the computed limit plus 10 mm. Values in red area will make the nozzle mount crash with the print plate, values in yellow area can make nozzle mount crash with the print plate mounts and for values in the green are the nozzle mount can move freely.

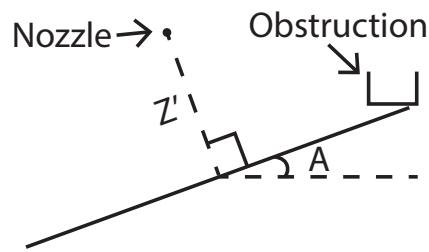


Figure 3.6: Illustration of how the relation between A-axis and Z' distance between the nozzle and the print plate are found.

3.3 Modeling the first version

3.3.1 The platform

When redesigning the platform, a number of methods to create the mounting between the rotary system and the platform were considered. The original system uses laser cut plywood, and the same would have been preferable on this system. But since ROBIN did not have access to a laser cutter when the first version was created and milling the material would both be expensive and time consuming. The parts were therefore designed to be printable with the Fortus 250mc. As mentioned in section 3.2 an extra motor has to be added to the platform. To have the same motors on the Y-axis, the original motor was removed and two motors of the type SM-42BYG011-25 were added. The motor specification for this motor can be seen in Section 3.1.4. As seen in Figure 3.3 these motors are connected to separate drivers. This was done because there were unused drivers on the system, and to avoid problems with parallel and serial connection of motors [54]. The design of the new platform was inspired by the old design. The parts of the original design were mirrored and then welded together with a distance between them. To make sure the platform was rigid, a support structure was created in the bottom of the platform to stabilize the aluminium profiles. The result of the new platform compared to the Ormerod platform can be seen in Figure 3.7.

3.3.2 The rotary system

As mentioned in section 3.2, the print plate should be in focus when designing the rotary system. The system should also have the ability to calibrate the print plate to make it intersect with the rotary axes. To be able to do this, the mounting for the print plate consists of three points which can easily be adjusted, as seen in Figure 3.8. A mount for the stepper motor and the worm gear system was made. The worm gear system for the C-axis mount can be seen in Figure 3.9. In this system the worm gear is connected to an axle which rotates the C-axis. The worm is connected to the stepper motor which then interacts with the worm gear and transforms the rotation from the motor to the C-axis. As mentioned in section 3.2 the worm and the motor axle had different diameters, therefore a lining with 0.5 mm thick walls was made, as seen in Figure 3.10. Considering this part had to be precise with 0.5 mm thick walls, it was decided to print this part with the Connex 500, which has a higher accuracy than the Fortus 250mc. The axle that connects the worm gear with the mount for the print plate, was connected to this mount with the help of two bearings and a link. The link has two screws tightened around the axle, as seen in Figure 3.11. When the C-axis was modulated, a mount for the C-axis was created so that the C-axis was able to rotate freely from -90° to 90° around the A-axis, this can be seen in Figure 3.12. A motor mount similar to the one on the C-axis mount was added to the A-axis mount, as a result the system was able to rotate around the A axis.



Figure 3.7: The evolution of the platform. The top picture are the platform of the original version (Ormerod), and the bottom are the platform of the first version



Figure 3.8: Print plate attachment on the first version.

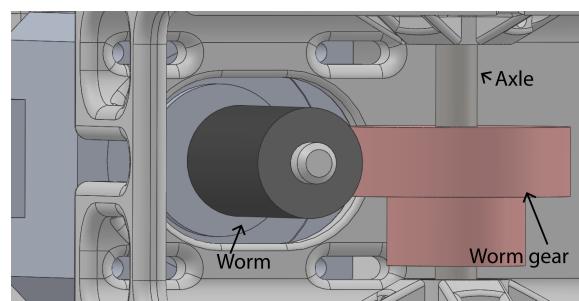


Figure 3.9: Motor and worm gear system on the C-axis mount, for the first version.



Figure 3.10: Motor axle bearing.

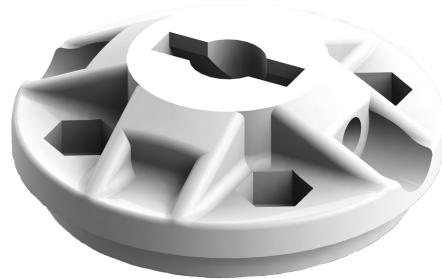


Figure 3.11: Axe to rotary system link, for the first version.

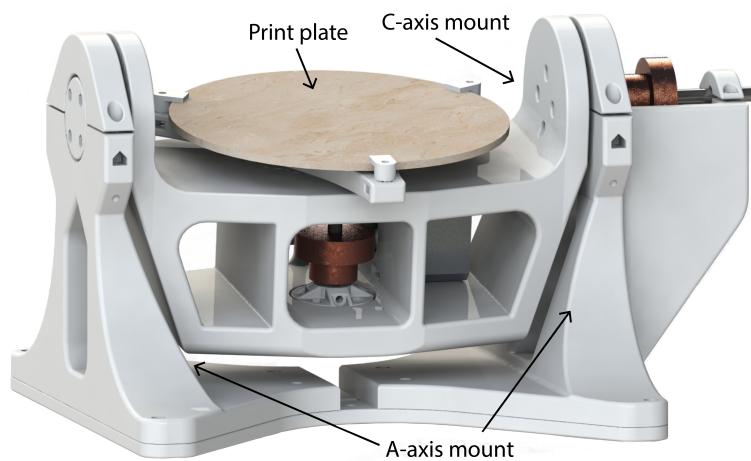


Figure 3.12: Rotary system on the first version.

3.3.3 Lowering the nozzle

As mentioned in section 3.2, the nozzle mount on the original system gave the system severe limits regarding the relation between the A-axis angle and the Z' distance between the nozzle and the print plate. To compensate for this, the mount for the nozzle was lowered by 50 mm, as seen in Figure 3.13. By doing this, the system has the possibility to reach a higher A-axis angle with a smaller Z' value, as seen in Figure 3.14. The new nozzle mount as illustrated is much bigger than the previous versions, and because of this the X-axis had to be extended so the X-axis sensor would still work, the result of this can be seen in Figure 3.15.

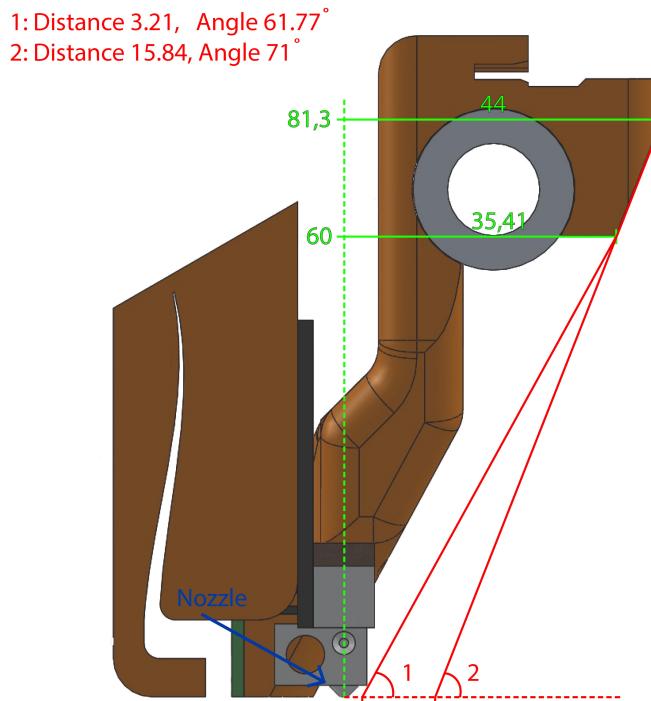


Figure 3.13: First version nozzle mount, all distances are in mm.

3.3.4 The Z-axis

When adding the rotary system on top of the Y-axis, the print plate will be raised as mentioned in section 3.2. The height will be raised about 120 mm combined with a 50 mm lowered nozzle, removes 170 mm of the 200 mm Z-axis range. Due to this, the Z-axis had to be extended. This was done by replacing the aluminium extrusion, the smooth rod, and the M5 threaded rod, that are 350 mm with identical parts that are 600 mm. With these parts attached the printer has a 250 mm extended Z-axis, and gives a range of 280 mm with the rotary system and the lowered nozzle. To make sure the M5 threaded rod were not wiggling too much, due to its expanded length, the top mounting for the Z-axis where expanded to keep the M5 threaded rod in place, as seen in Figure 3.16.

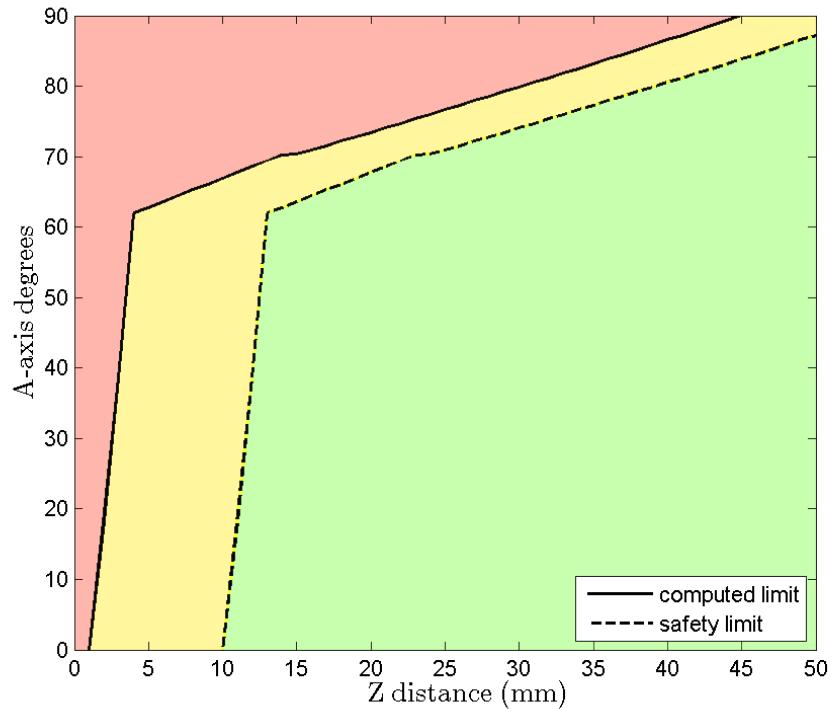


Figure 3.14: A-axis and Z' limits for First version. Computed limit is found as mentioned in section 3.2 and safety limit is the computed limit plus 10 mm. Values in red area will make the nozzle mount crash with the print plate, values in yellow area can make nozzle mount crash with the print plate mounts and for values in the green are the nozzle mount can move freely.

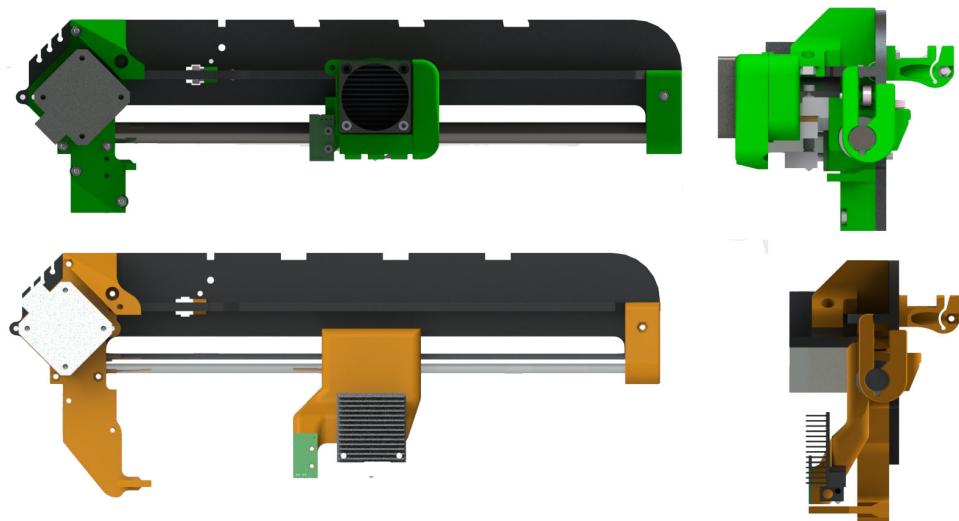


Figure 3.15: The evolution of the X-axis. The top picture are the X-axis of the Ormerod, and the bottom are the X-axis of the first version.

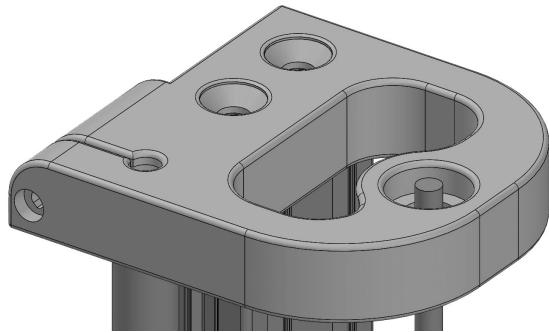


Figure 3.16: Top mount for Z-axis.

3.4 Modeling result of the first version

The first modeling process resulted in the first version, as seen in Figure 3.17. When assembling the first version, some problems with the initial design were noticed. One of the major problems was that the screw positions for the motors on A- and C-axis were difficult to reach. Because of this, it was difficult to adjust the distance between the worm and the worm gear. When the distance between the worm and the worm gear could not be adjusted properly, the worm gear system did not function accurately, and the A- and C-axis ended up with a backlash of $15^\circ - 26^\circ$. The backlash was not a problem for the A-axis since it was limited to $0^\circ - 90^\circ$, and therefore preloaded by the weight of the C-axis mount. However, for the C-axis this would be a severe problem when the rotation changes direction. To compensate this, the C-axis were preloaded to make sure the backlash was minimized. When the system was preloaded other problems surfaced, due to the added weight from the preload, the Y-axis was missing steps when moving. To compensate this the speed of the Y-axis was reduced until the Y-axis was able to move without losing steps.

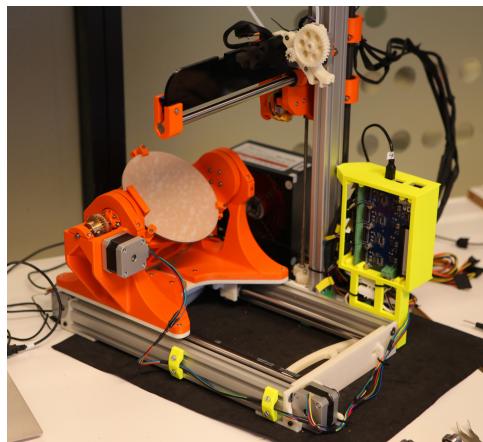


Figure 3.17: The first version of the 5-axis FDM printer.

3.5 Testing the first version

From the beginning, it was clear that the system had some errors. To figure out the reason for this, both the axes were run 0° to 90° with steps of 1° three times each. A digital inclinometer was used as a method to efficiently find which precise angle the axes approached. The digital inclinometer works as a leveler, but allows for the exact measurement of an angle. The digital inclinometer used in these tests have an accuracy of $\pm 0.2^\circ$ [36]. Results from these tests can be seen in Figure 3.18 and Figure 3.19. These graphs shows the median of errors from the tests and range of the samples. Both the plots show the error of the axis along the X-axis and the degrees and revolutions of the worm along the Y-axis.

As seen in the plots, the error of the first version are quite severe. When considering the plot and the revolutions of the motor, it is obvious that the error is periodic. The error comes from the way the worm was connected to the stepper motor. The lining between the stepper motor and the worm was not working as it should, and therefore the worm had an elliptic movement. This elliptic movement made the teeth of the worm and the teeth of the worm gear to have different distances, depending on where the motor was in a revolution. The worm would then have a changing effect on the worm gear.

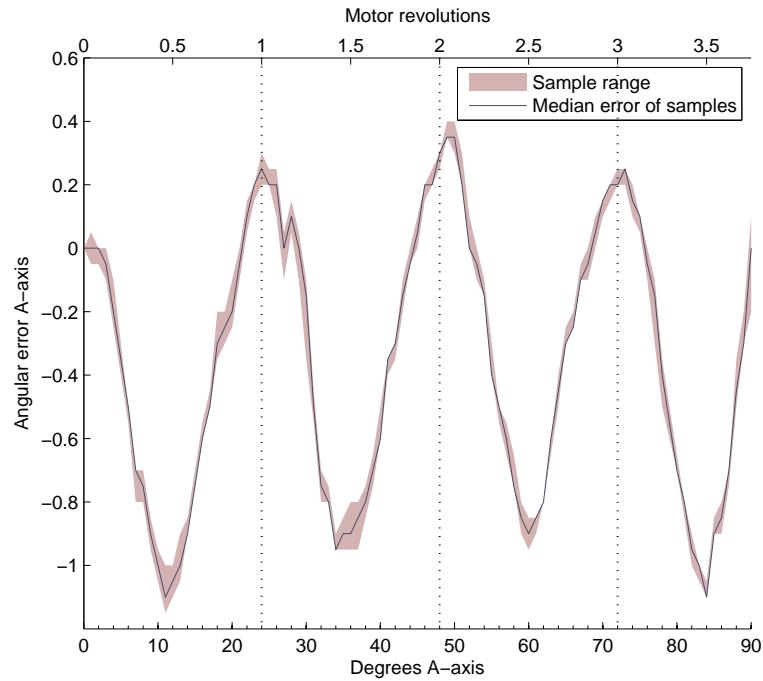


Figure 3.18: Angular error A-axis, first version.

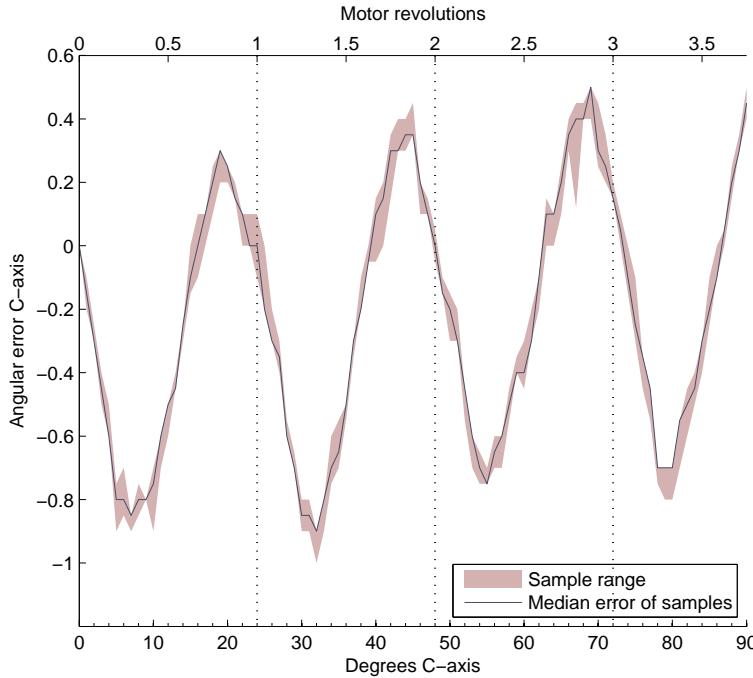


Figure 3.19: Angular error C-axis, first version

3.6 Problems with the first version

After working with the first version for a while, more problems appeared. The plastic around the screws on the platform cracked because the plastic was too thin. The links that connected the axle of the worm gear to the A- and C-axes were too thin and cracked which resulted in more backlash for the A- and C-axes.

Because of the problems that have been described, specifically the movement problems of A- and C-axis, the first version was not able to print properly with the 5-axis system. Therefore it was decided to modulate a new version where these problems were accounted for.

3.7 Planning the second version

The first version showed that the initial design had some problems with applying a good distance between the worm and the worm gear. One way to compensate for this problem was to create a motor and worm mount where the distance between the worm and the worm gear could be adjusted. The first version also had problems with the strength of the mounting points for the platform and could be improved by simply creating a thicker wall where the platform was mounted. The nozzle on the first version still created a limitation for the relation between Z' and the angle of the A-axis, as can be seen in Figure 3.14. Trying to remove this limitation the nozzle had to be redesigned. Two of the possible improvements were to lower the nozzle

even more, and remove as much of the structure as possible on one side of the nozzle.

While planning the new rotary system it was apparent that the new design would increase in size and thereby mass. To make sure both the A-axis and the Y-axis would be able to move the new system, new and stronger motors were ordered. The specification of the new motors, 42BYGHM809, can be seen in Section 3.1.4. The new motors have doubled accuracy, 400 steps each revolution compared to 200 steps each revolution, and a holding torque that are more than double the original motors. Because of the doubled accuracy, the new system has an improved worst-case accuracy for the Y- and A-axis, as seen in Section 3.7.

Axis	$\frac{\text{step}}{\text{mm}}$	$\frac{\text{mm}}{\text{step}}$
X	87.489	0.01147
Y	174.978	0.01147
Z	4000	0.00025
A	218.34	0.00458
C	109.17	0.00916

Table 3.3: Motor accuracy in the worst case for the second version

3.8 Modeling the second version

3.8.1 The motor mount

To have a system that can adjust the distance between the worm and worm gear, the motor mount on the second version was connected with a sliding mount. This can be adjusted with the help of two screws and two springs between the A- or C- axis mount and the motor mount, as seen in Figure 3.21. Thus it is possible to adjust the distance between the worm and the worm gear. To deal with the size difference between the hole of the worm and the motor axle, the worm are attached to a 6 mm rod, which then were connected to the motor as seen in Figure 3.20. This way, the worm will not be able to move in an elliptic trajectory, and therefore the movement problems from the first version were eliminated. The motor mount was modulated so that the same part could be used for the A- and C-axis.

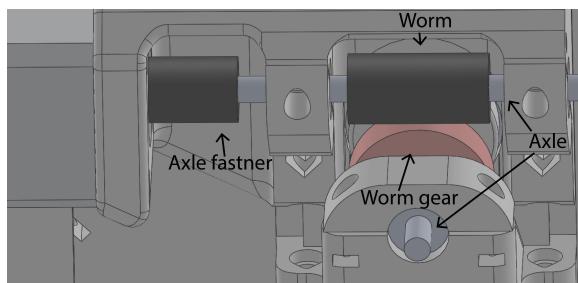


Figure 3.20: Motor and worm gear system on C-axis for the second version

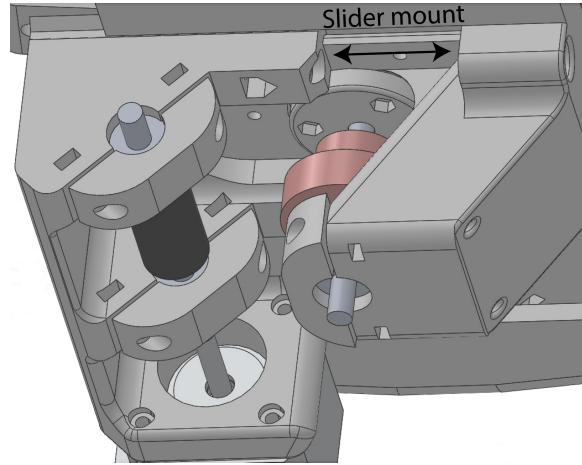


Figure 3.21: Sliding motor mount on the second version

3.8.2 The rotary system

Because of the new motor mount, both the A-axis mount and the C-axis mount had to increase in size. To simplify the modeling process, the C-axis mount were designed first. When the design of the C-axis mount were finished, the A-axis mount was quite straightforward. The result of the rotary system can be seen in Figure 3.22. In the first version, the links that connected the axle of the worm gear to the A- and C- axis broke. In order to compensate for this, the rod was extended to go through the bearing and fastened to the rotating parts. By doing this, it was easier to create a more durable fastening system. Due to this the print plate mount had to be expanded. While redesigning this a more accurate system for calibrating the print plate also were added. This calibrating system uses three springs, which are between the print plate mount and the holders of the print plate.

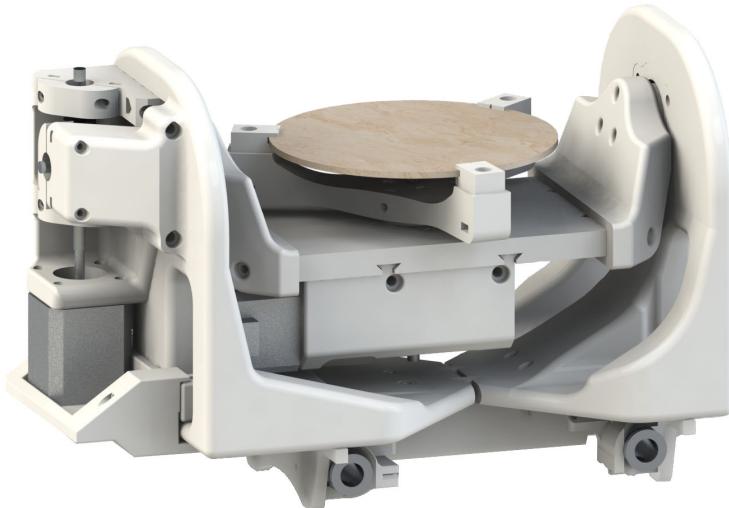


Figure 3.22: Rotary system on the second version

3.8.3 The platform

As mentioned in section 3.7 one of the problems in the first version were that the frame around the screws on the platform were breaking. To avoid this problem the thickness of the platform had to be expanded. The new motors that are driving the Y-axis, are larger than the original motors, as can be seen in Section 3.1.4. Due to this the motor mount on the platform had to be remade to accommodate this.

During the modeling of the platform, the ROBIN group got access to a laser cutter, which allowed a more rigid redesign of the mounting between the rotary system and the platform. The new platform with the acrylic parts can be seen in Figure 3.23.



Figure 3.23: The platform of the second version

3.8.4 Lowering the nozzle even more

As mentioned in section 3.7, even when the nozzle was lowered in the first version, there were still problems to print lower layers with an angle approaching 90° on the A-axis, as seen in Figure 3.14. To reduce this limit in the system, a new nozzle mount was designed. The nozzle was lowered even more and plastic from the backside of the nozzle was removed. The result of this can be seen in Figure 3.24. The new nozzle mount are as seen much bigger than the previous versions. Because of this, the X-axis had to be extended to compensate for the extra weight and the lower X-axis sensor. The result of this can be seen in Figure 3.25. Figure 3.26 show the result of the new nozzle mount. The new system can reach 90° already at 20 mm if the safety limits are considered. Where the Ormerod's and First version's nozzle mount limited the system, they were not able to reach 90° before the distance was over 50 mm, if the safety limit were considered.

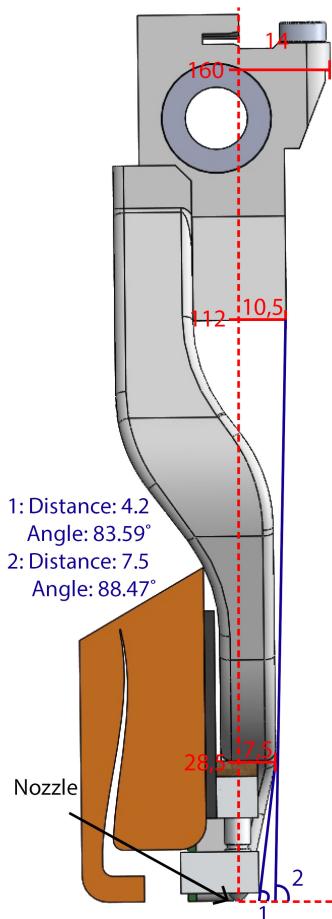


Figure 3.24: The nozzle mount of the second version nozzle, all distances are in mm.

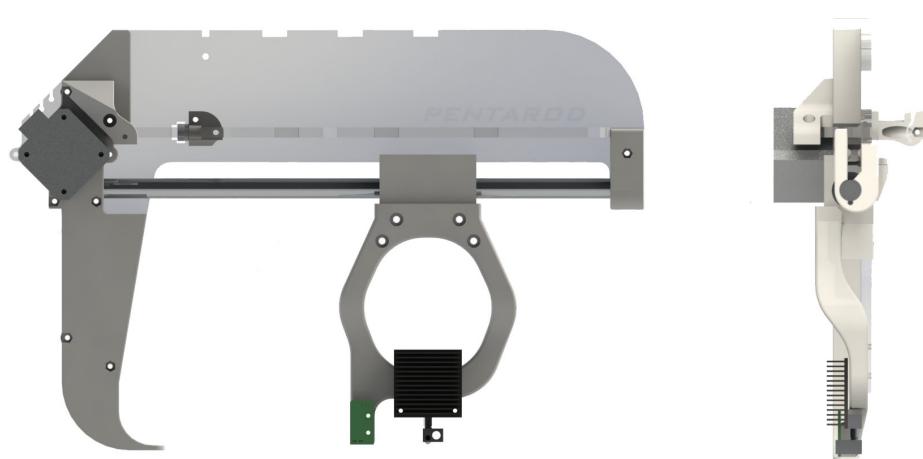


Figure 3.25: The X-axis of the second version

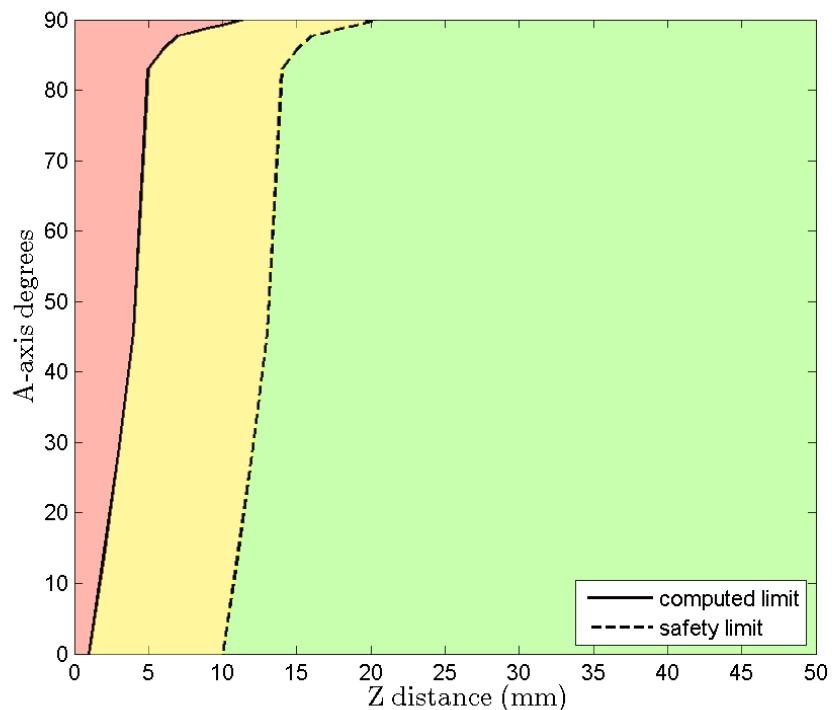


Figure 3.26: A-axis and Z' limits for the second version. Computed limit is found as mentioned in section 3.2 and safety limit is the computed limit plus 10 mm. Values in red area will make the nozzle mount crash with the print plate, values in yellow area can make nozzle mount crash with the print plate mounts and for values in the green are the nozzle mount can move freely.

3.9 Modeling result of the second version

The result of the second version, called Pentarod, can be seen in Figure 3.27. When assembling the second version, a problem with the sliding system for the motor holder appeared. While calibrating the distance between the worm and the worm gear, it still was difficult to find a perfect distance between the worm and the worm gear. To close together the motors would not be able to rotate the gears. To far apart, the system would have too much backlash. However, with enough calibration the backlash in the A- and C-axis was reduced to $1^\circ - 2^\circ$.

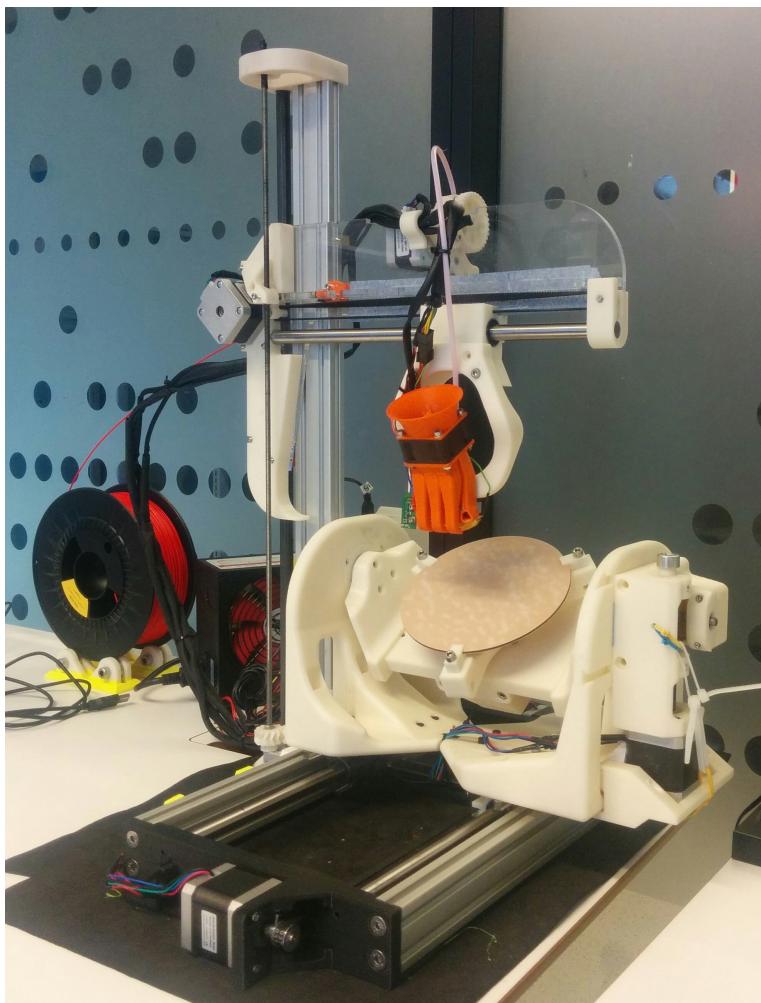


Figure 3.27: The second version of the 5-axis FDM printer.

3.10 Testing the second version

The second version was tested the same way as the first version, both the axis was moved from 0° to 90° with 1° step three times. The result of this can be seen in Figure 3.28 and Figure 3.29, which shows the median of the errors from the tests and the range of the samples. These plots shows that the new system has an improved accuracy for the A and C axis movement. There are still an angular error of -0.15° to 0.15° in the system, and since these values are well within the accuracy of the digital inclinometer, $\pm 0.2^\circ$, as mentioned in section 3.5, this was considered acceptable.

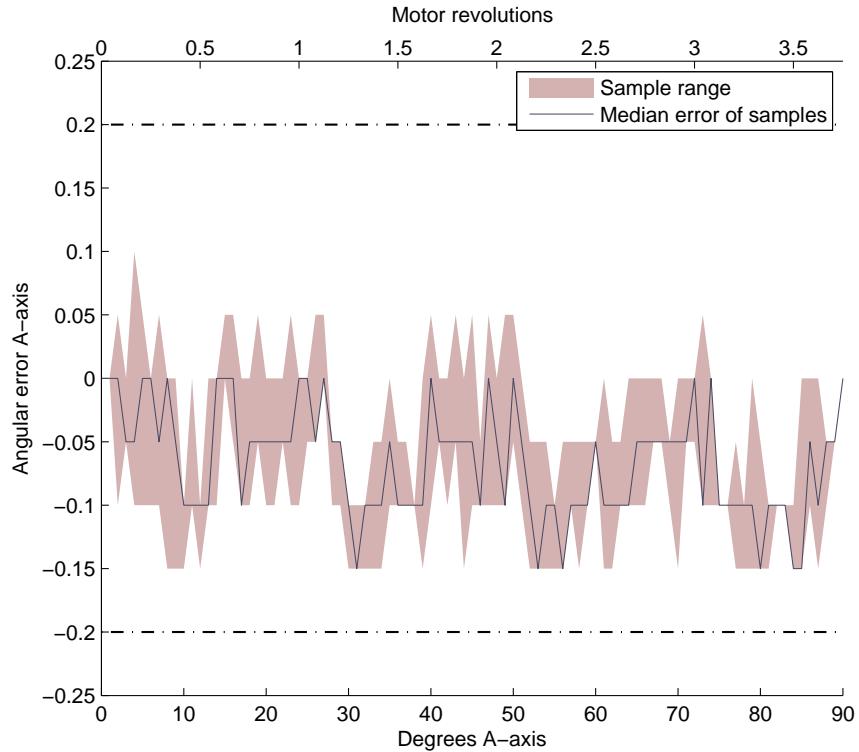


Figure 3.28: Angular error A-axis of the second version.

To make sure this system was rigid enough to be used over time, the system was stress tested by running the sequence, shown in the G-codes 3.1 and 3.2, 15 times. The result of running this sequence on the A-axis four times can be seen in Section 3.10. As shown the A-axis still have some angular error after a long run. Since the angular error are small, specifically if the digital inclinometer accuracy of $\pm 0.2^\circ$ are taken into account, this error were considered acceptable for testing the system. The result of running this sequence on the C-axis four times can be seen in Section 3.10. As illustrated the angular error of the C-axis are minimal and well within the angular error already obtained from the previous test.

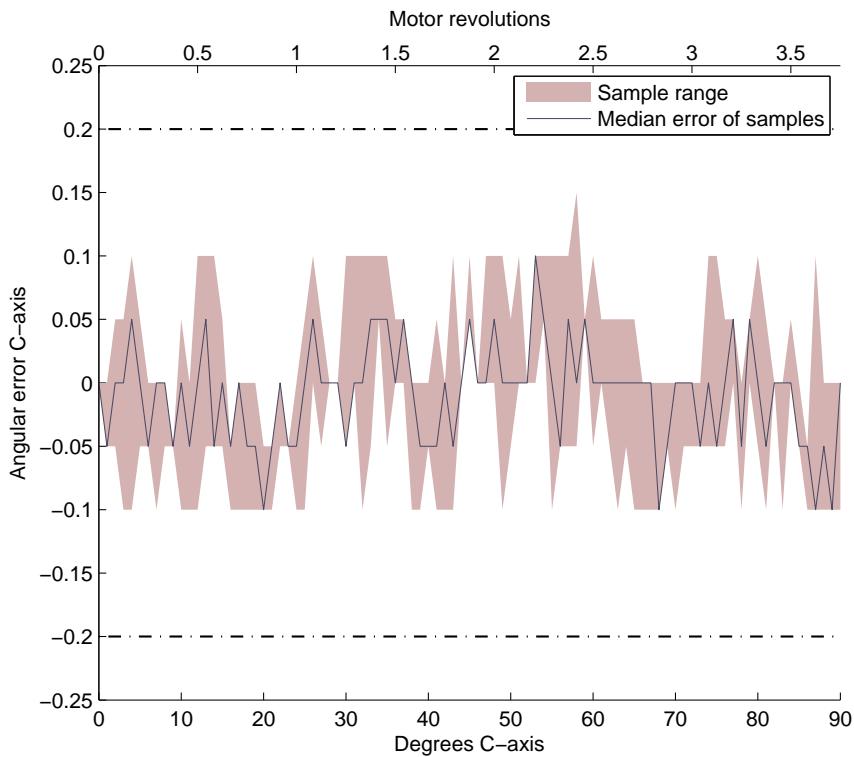


Figure 3.29: Angular error C-axis of the second version.

Should be	Error after run			
	0°	10°	45°	90°
1. run	-0.25°	9.6°	44.6°	89.8°
2. run	-0.4°	9.6°	44.5°	89.75°
3. run	-0.4°	9.5°	44.5°	89.8°
4. run	-0.4°	9.5°	44.5°	89.75°

Table 3.4: Results of A-axis stress test for the second version

	Error after run			
	0°	90°	180°	360°
1. run	-0.05°	90°	180.1°	360°
2. run	0°	90.05°	179.95°	359.9°
3. run	0°	89.95°	180°	359.95°
4. run	0.05°	90°	180.05°	360.05°

Table 3.5: Results of C-axis stress test for the second version

```
1 G1 A0
2 G1 A45
3 G1 A90
4 G1 A45
5 G1 A90
6 G1 A45
7 G1 A0
8 G1 A45
9 G1 A0
10 G1 A90
```

G-code 3.1: Stress test A-axis.

```
1 G1 C0
2 G1 C180
3 G1 C360
4 G1 C180
5 G1 C360
6 G1 C180
7 G1 C0
8 G1 C180
9 G1 C0
10 G1 C360
```

G-code 3.2: Stress test C-axis.

Chapter 4

Programming a 5-axis system

This chapter presents how the original firmware is working and how it is modified to work as a 5-axis system.

4.1 Original system

It was decided in the early stages of the project to use an already working firmware for the 3-axis 3D printer instead of creating a new system for 3D printing. Hence it was possible to concentrate on the additional two axes. At the beginning of this thesis, there were three main versions of the Ormerod firmware: the original, ZPL's version and DC42's version. The three versions of this firmware could easily been used for this project, but the DC42 version had the best reviews by the community and therefore this version was chosen.

4.2 5-axis system

As mentioned in section 2.9 5-axes 3D printing with plastic did not exists in the beginning of this thesis. Consequently the inspiration comes from 5-axes CNC. This was an obvious field to look into when deciding how the system should interpret the G-code and the format the G-code. As mentioned in section 2.5.1, one of the ways the 5-axis CNC represents its cutter location (CL) is by using CL points represented in ISO format as {X, Y, Z, I, J, K}. In this ISO format the {X, Y, Z} are the coordinates of the CL, and {I, J, K} are the direction values of the tool vector [43]. When using this G-code format, it is assumed that the part is fixed, and that the cutter completes all the movements. This makes it possible to create G-code which would work on different 5-axis machines as long the machine translates the G-code to work with its own setup [6]. When the part is fixed, the G-code are also easier to comprehend and therefore easier to manipulate or manually write.

To be able to implement this system, the original code, which has three drivers for the axes and five axes for the extruder's, had to be rewritten to have five drivers for the axes, X, Y0, Y1, Z, A and C. This was done by changing a few parameters in the original code. The steps per unit that were used are shown in Table 4.1 for v1, the first version and v2, the second version.

Axis	X	Y	Z	A	C
Steps per unit v1	87.4890	87.4890	4000.0	133.3333	133.3333
Steps per unit v2	87.4890	174.9780	4000.0	267.6666	133.3333

Table 4.1: Motor steps per unit.

X, Y and Z are in steps per mm and A and C are in steps per degree.

4.3 Tool vector to angles

For the system to interpret the G-code, it has to compute the A and C angles from the values {I, J, K} of the tool vector. To find the angles of the tool vector it is possible to look at these values for the tool vector as a Cartesian coordinate representation of a spherical coordinate system, as shown in Figure 4.1, where $I = X$, $J = Y$ and $K = Z$. Because the tool vector length always should be equal one in the G-code [5] it is possible to find the A and C angles by using the relation between Cartesian coordinates and spherical polar coordinates as shown in eq. (4.1) [7, 60]. By setting $X = I$, $Y = J$, $Z = K$ and $r = 1$, eq. (4.1b) and (4.1c) becomes eq. (4.2a) and (4.2b). Where ϕ is the angle of the A-axis and θ is the angle of the C-axis. Since this system has the possibility to get $J = 0$ and a complete range of a whole circle is desired, instead of $-90^\circ \leq A \geq 90^\circ$, the atan2 function [17] is being used to find θ , as shown in Equation (4.2c) [23]. As mentioned in section 3.2 the A-axis is limited to $0^\circ - 90^\circ$, with this the I, J and K values get the limitations shown in Equation (4.3). These limitations for {I, J, K} and the functions used to find the angles, the system get the limits, $0^\circ \leq A \geq 90^\circ$, and $-180^\circ \leq C \geq 180^\circ$.

$$r = \sqrt{X^2 + Y^2 + Z^2} \quad (4.1a)$$

$$\phi = \arccos\left(\frac{Z}{r}\right) \quad (4.1b)$$

$$\theta = \arctan\left(\frac{Y}{X}\right) \quad (4.1c)$$

$$\phi = \arccos(K) \quad (4.2a)$$

$$\theta = \arctan\left(\frac{J}{I}\right) \quad (4.2b)$$

$$\theta = \text{atan2}(J, I) \quad (4.2c)$$

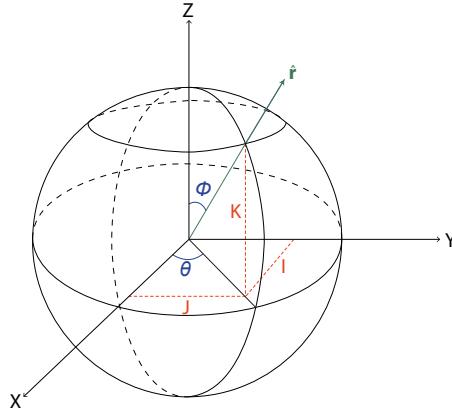


Figure 4.1: Spherical polar coordinates visualization. Figure based on figure from Wolfram MathWorld [60]

$$-1 \leq I \leq 1 \quad (4.3a)$$

$$-1 \leq J \leq 1 \quad (4.3b)$$

$$0 \leq K \leq 1 \quad (4.3c)$$

4.4 Transformation

This system has two rotary axis that rotate around the X-axis and the Z-axis. The transformation matrices in eq. (4.4) and (4.5) [53] give a rotation transformation around X and Z. With these, the system is able to transform the G-code format from a coordinate system with a fixed part, to a system with a moving part. This transformation can be seen in Figure 4.2 for the A-axis and Figure 4.3 for the C-axis.

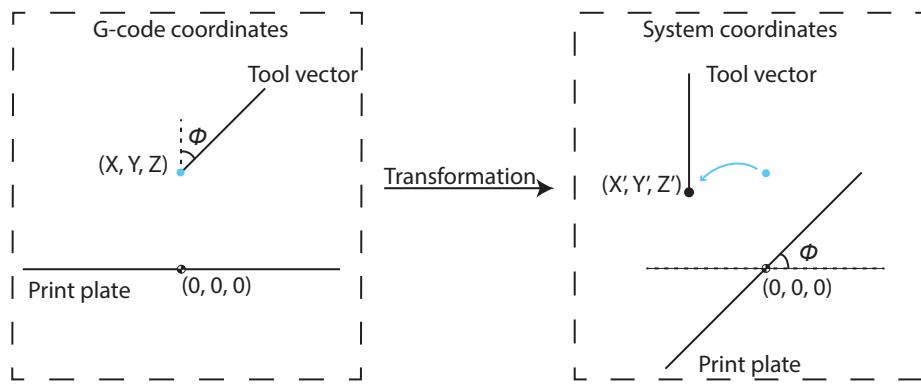


Figure 4.2: Illustration of the transformation along A-axis, A is represented by {I, J, K}.

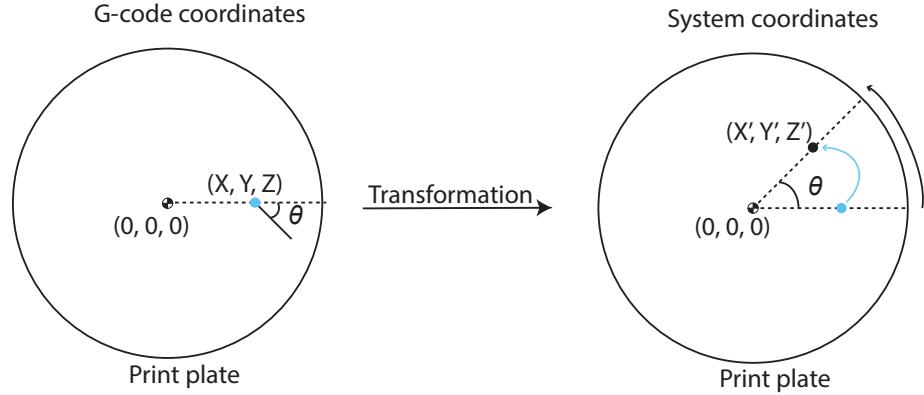


Figure 4.3: Illustration of the transformation along C-axis, C is represented by {I, J, K}.

$$R_{(X,A)}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \quad (4.4)$$

$$R_{(Z,C)}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.5)$$

The transformation matrix is found by multiplying these matrices, as shown in eq. (4.6). Then the transformed coordinates {X', Y', Z'} can be found by multiplying the transformation matrix by the original coordinates {X, Y, Z}, shown in Equation (4.7).

$$T = R_{(X,A)}(\phi) * R_{(Z,C)}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \cos(\phi) * \sin(\theta) & \cos(\phi) \cos(\theta) & -\sin(\phi) \\ \sin(\phi) * \sin(\theta) & \sin(\phi) \cos(\theta) & \cos(\phi) \end{bmatrix} \quad (4.6)$$

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = T * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (4.7)$$

4.5 Coding the 5-axis system

When the original system receives a G-code, it looks at the first entry to see which type of command it receives, which can be seen as the “G-code handler” in Figure 4.4. If the system gets a G1 command, it reads the rest of the G-code and put new values for X, Y and Z into the *moveBuffer* array, which is done by the “G1 parser” in Figure 4.4. The step function, “G1 to step” in the figure, then reads the *moveBuffer* array to figure out how many steps each of the motors need to move based on Table 4.1. When adding three motors, an extra for Y, A and C, the *moveBuffer* array is being changed from {X, Y, Z, E0, E1, E2, E3, E4} to {X, Y0, Y1, Z, A, C, E0, E1}. By rewriting the “G1 parser” the system reads in the G-code normally. In addition it also sets values from Y into Y0 and Y1, and values from A into A and C into C. With these modifications, the system is capable of moving all the axes.

When working with a 5-axis system the G-code has to be translated to work with the system. To do this two functions have been added to the code, “Vector handler” and “Transformation handler”, as shown in Figure 4.4. The two functions can be seen in Pseudo code 4.1, for the “Transformation handler”, and Pseudo code 4.2, for the “Vector handler”. To be able to use the system as a 3-axis system, and to be able to use both the G-code format mentioned in section 2.5.1, “G1 X Y Z A C F E” and “G1 X Y Z I J K F E”. Two G-code commands have been added to the system, G43, which only turn on the Transformation handler, and G44, which turns on both the handlers. An illustration of how these works are shown in Figure 4.4. In these G-codes X, Y and Z are the coordinates, A and C are the angles, I, J and K are the values for the tool vector, F is the feedrate and E is the movement of the extruder.

The vector handler computes the angles A and C from I, J and K with eq. (4.2a) and (4.2c). The A and C in this system are represented as degrees and since the C++ functions *acos* and *atan2* returns radians [16, 17], the returned values are multiplied with $\frac{180}{\pi}$ to get degrees. Since *atan2* get a domain error in C++ when both I and J are 0 [17], these cases are skipped and the last known value of C are stored in C. This case is shown in lines 11 and 23-25 in Pseudo code 4.2. The function *atan2* is also limited to $[-\pi, +\pi]$ in C++ [17], thusly the difference between the last known value of C and the new C are checked. If the movement is higher than 180° , 2π is added to the return value of *atan2* before it are multiplied with $\frac{180}{\pi}$. To control whether 2π or -2π have to be added to the return value, or if the C-axis already have rotated multiple times, a variable *PIFacton* counts how many rotations the system already have in either negative or positive direction. This case is shown on the lines 14-20 in Pseudo code 4.2. An illustration of this case can be seen in Figure 4.5, where the last G-code is at the start point and current G-code are at the end point, by using *atan2* without adding *PIFacton* * 2π , the system will interpret it as cases 1 and 2 and will move as case 4. By adding -2π as it is done in case 3 the system will move as case 5 instead, which is a more logical movement.

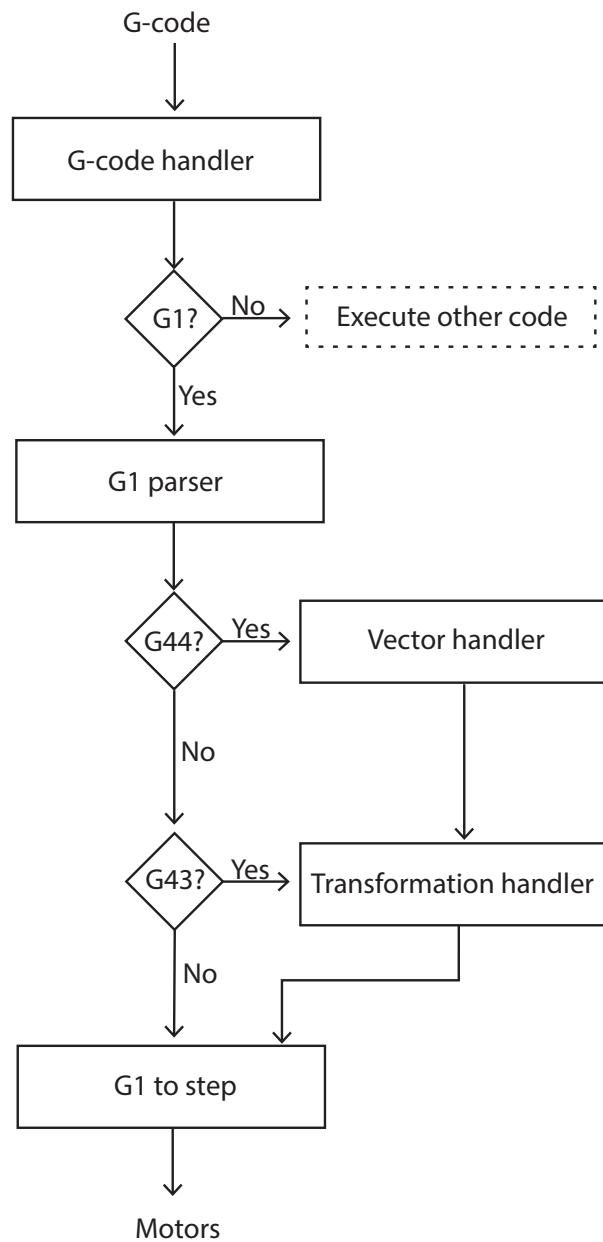


Figure 4.4: Illustration of how the system handles G-code

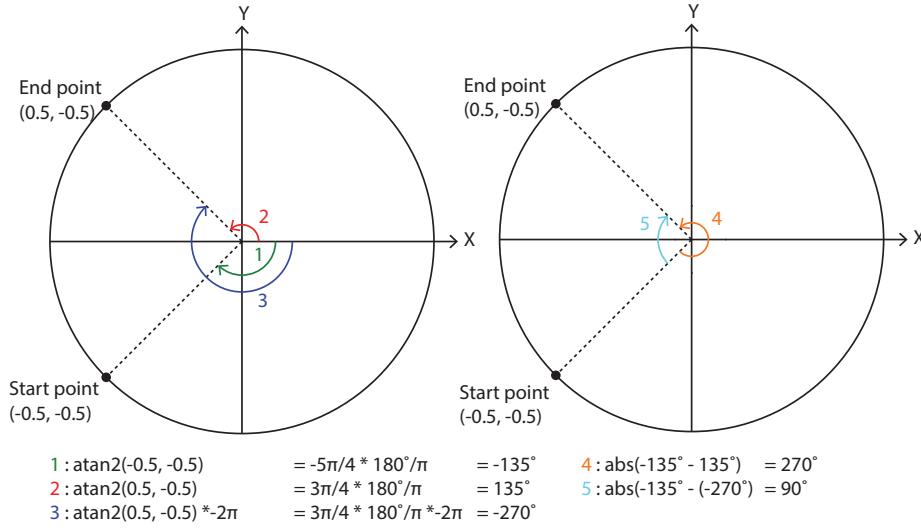


Figure 4.5: Illustration of C-axis movement

The transformation handler transform {X, Y, Z} from the G-code with Equation (4.7), (4.6) and the values of A and C. The transformation can be seen in line 10-12 in Pseudo code 4.1. Since the C++ functions *sin* and *cos* accepts only radians [18, 19], the angles are multiplied with $\frac{\pi}{180}$.

As mentioned in section 2.5, G-code uses the old values if new are not supplied. To be able to handle these cases the G-code values for X, Y and Z is stored in *posBuffer* array, which updates the values for X, Y and Z when they are changed in the G-code.

```

1 Require: moveBuffer with valid {A, C}
2 Require: posBuffer with valid {X, Y, Z}
3 Require: PI = 3.14159265
4
5 if (G43 or G43){
6
7     (Get {A, C} from moveBuffer);
8     (Get {X, Y, Z} from posBuffer);
9
10     $X^T = X * \cos(C * PI/180) - Y * \sin(C * PI/180);$ 
11     $Y^T = X * \cos(A * PI/180) * \sin(C * PI/180) + Y * \cos(A * PI/180) * \cos(C * PI/180) - Z * \sin(A * PI/180);$ 
12     $Z^T = X * \sin(A * PI/180) * \sin(C * PI/180) + Y * \sin(A * PI/180) * \cos(C * PI/180) + Z * \cos(A * PI/180);$ 
13
14    (Load {X',Y',Z'} into moveBuffer);
15 }
```

Pseudo code 4.1: X, Y, Z transformation.

```
1 Require: G-code string
2 Require: PI = 3.14159265
3 Require: PIFaction with valid number
4
5 if (G44)
6     (read {I, J, K} values from G-code into {I, J, K} buffer);
7
8 if ({I, J, K} values found) {
9     A=acos(K)*180/PI;
10
11    if (I != 0 and J != 0){
12        C=(atan2(J,I)+PIFaction*(2*PI))+180;
13
14        if (abs(C - lastC) > 180){
15            if (C - lastC < 0){
16                PIFaction=PIFaction+1;
17            }
18            else{
19                PIFaction=PIFaction-1;
20            }
21        }
22    }
23    else{
24        C = lastC;
25    }
26
27    lastC = C;
28    (Load new {A, C} into moveBuffer);
29}
30}
```

Pseudo code 4.2: Tool vector to angles.

Chapter 5

Printing with a 5-axis system

This chapter will present how the G-code for the test parts was created, how the printer was calibrated, and the printing process for these parts.

5.1 G-code

As mentioned in section 1.1, 3-axis FDM printers have many limitations because they are only able to print from bottom to top. When printing with a FDM printer it is considered a good practice to avoid using overhanging angles over 45° for a good quality print without using a support structure. There are printers which can print with good quality on even higher angles, but all of them would start to have problems when the angle of overhang approaches 90° . Another limitation for 3D printers using the FDM technology is as mentioned in section 1.2, that they are printing layer by layer. This creates a stair-like structures on surfaces that are not parallel with the build plate. Each step has the same height as the layer height, as seen in Figure 1.1. This can be avoided to some extent by having a low layer height, but this is time consuming. A 5-axis system should, on the other hand, have no problems with these limitations. Therefore, two structures which can simulate these limitations to some degree were chosen as tests for the 5-axis 3D printer. To create something easy to work with, but also complex for the printer, a cylinder with a spherical cap was used for testing the surface limitation. In addition, a round structure with a 90° overhanging top was created to test the overhang limitation. Both of these can be seen in Figure 5.1. The two objects have a distance between the print plate and the 5-axis print, which is 30 mm for the surface test and 40 mm for the support test. This is necessary because the size of the nozzle makes it difficult to reach the print plate on higher A-axis angles, as mentioned in Section 3.3.3 and 3.8.4. Both of the distances are higher than necessary to avoid crashes between the nozzle and the rotary system in the case of an unexpected error in the system or the G-code. For the surface test two strategies to create the path were used; one where the paths are in the X- and Y-axes, and one where the paths are in the X- and Z-axes.



Figure 5.1: Test structures: surface test to the left and support test to the right

5.1.1 Creating G-code

As mentioned in section 4.2 the G-code format is either “G1 X Y Z A C F E” or “G1 X Y Z I J K F E” for the 5-axis system. To create G-code with this format HSMWorks multi-axis function was first tested, which also can simulate the G-code on the 3D printer, as shown in Figure 5.2. Using this simulation was a great way of testing the system and figuring out the limits of the axes. In HSMWork it is possible to save the G-code as Fanuc TCP type II which has the format “Go1 X Y Z I J K F”. Since the G-code is for CNC mills it had to be rewritten to work with this system. This was done by rewriting the syntax, removing CNC specific code, adding printer code and adding extruder values. The extruder values used were found by finding the distance from previous point to current point and multiplying it with a plastic extrusion value and the layer height of the print. The plastic extrusion value (0.2078760) was found by analyzing working G-code files for 3-axis 3D print system with a Java program. Pseudo code is shown in 5.1. The method goes through the G-code and finds the distance between current point and previous point, it then divides the current extrusion value by this distance and saves the found extrusion value in a buffer. When G-code methods are used it analyzes the buffer with the extrusion values, and finds the highest occurring value. This is then divided by the layer height of the G-code.

Working with HSMWorks to create G-code proved to be limiting because the system is made to remove material not add it, and creating support structure using HSMWorks proved to be difficult. Therefore it was decided to create a Java program which could generate G-code for a 5-axis system.

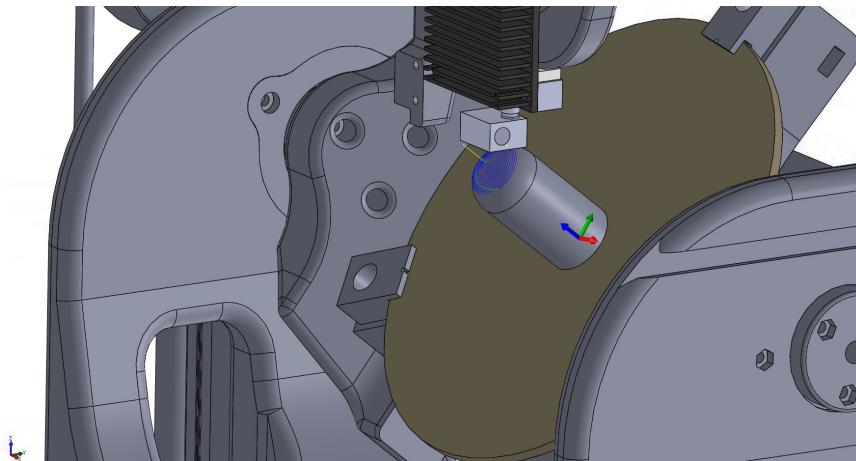


Figure 5.2: G-code simulation in HSMWorks

```
1 (Read all G1 G-code file into buffer);
2 pre_g = buffer.next;
3
4 while(buffer.hasNext){
5     cur_g = buffer.next;
6     if(cur_g has e value){
7         dist = (distance between cur_g and pre_g);
8         eBuffer = cur_g.e/dist;
9     }
10    else{
11        pre_g = cur_g;
12    }
13 }
14
15 e = (Analyze eBuffer to find highest occurring e)
16 print(e/layer height in the file)
```

Pseudo code 5.1: Computing extruder value from G-code files.

To find the direction values of the tool vector, spherical polar coordinates was used. With spherical polar coordinates it is possible to find the angles onto the surface of the sphere, based on the X, Y, Z coordinates, as Figure 4.1 shows. When the angles are found, the direction values of the tool vector will be the same as the unit vector of the radius, as shown in Equation (5.1) [7, 60]. The values for I, J, K can then be found with Equation (5.2). The methods to find these values can be seen in Java code 5.2.

$$\hat{r} \equiv \frac{\frac{dr}{d\theta}}{\left| \frac{dr}{d\theta} \right|} = \begin{bmatrix} \cos(\theta) * \sin(\phi) \\ \sin(\theta) * \sin(\phi) \\ \cos(\phi) \end{bmatrix} \quad (5.1)$$

$$I = \cos(\theta) * \sin(\phi) \quad (5.2a)$$

$$J = \sin(\theta) * \sin(\phi) \quad (5.2b)$$

$$K = \cos(\phi) \quad (5.2c)$$

To be able to create the G-code for the surface test, two mathematical methods was used; arc length [9] and spherical cap [59]. The arc length was used to find the A- and C- axis movement based on the resolution the print should have between two points and are shown in eq. (5.3). l is the resolution, r is the radius of the circle and θ is the angle between each circle or each point in the G-code. The spherical cap was used to find the radius and distance to the origin for the smaller circles, these are shown in eq. (5.4). a is the radius and $R - h$ is the distance to the origin of the circle. An illustration of this is shown in Figure 5.3. The helping method to find these values can be seen in Java code 5.3.

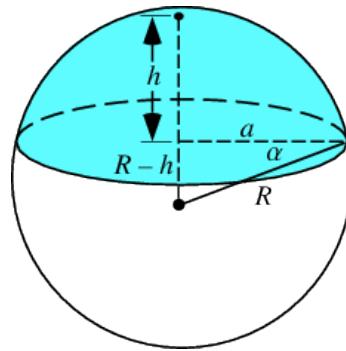


Figure 5.3: Visualization of spherical cap. The figure comes from Wolfram MathWorld [59]

$$\theta = \frac{l * 180^\circ}{r * \pi} \quad (5.3)$$

$$R - h = R * \sin(\alpha) \quad (5.4a)$$

$$a = \sqrt{h(2 * R - h)} \quad (5.4b)$$

For creating the complete 5-axis G-code three Java methods was made: method for creating spherical surface along X- and Y-axis, method for creating spherical surface along X- and Z-axis and method for creating circles along the X- and Y-axis. These methods utilize Java methods 5.2 and 5.3 to find coordinates and unit vectors for points along a spherical surface or a circle, and generates the G-code for each of these points. The three Java methods and their associated methods are shown in Appendix A. When the 5-axis G-code was generated, it was pasted into the end of a G-code file of the base structures for the surface tests. And between the base structure and the top structure for the support test. The surface tests are shown in Figure 5.1 and Figure 6.11 for the support test. The G-code for these structures were generated with a standard 2.5D slicer for 3D printers.

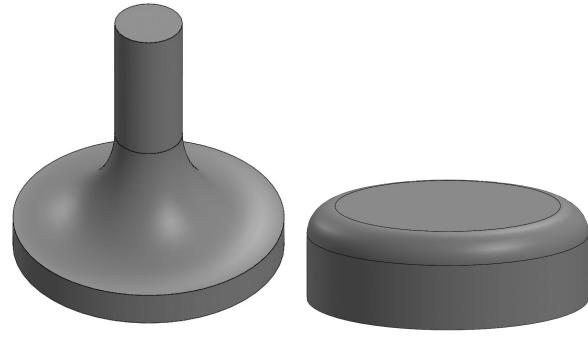


Figure 5.4: The parts of the support test

```

1  /**
2 *   Method to find vector values for a point in a sphere based
3 *   on the spherical angles to that point.
4 *
5 *   @param a           A-axis angle
6 *   @param c           C-axis angle
7 *   @return double[]  Vector values :
8 *                   0: I
9 *                   1: J
10 *                  2: K
11 */
12 private double[] cartAngleToVector(double a, double c){
13     double[] tmp = new double[3];
14
15     tmp[0] = cos(toRadians(c))*sin(toRadians(a));
16     tmp[1] = sin(toRadians(c))*sin(toRadians(a));
17     tmp[2] = cos(toRadians(a));
18     return tmp;
19 }
20
21 /**
22 *   Method to find angles and radius to a point in a sphere based
23 *   on the cartesian coordinates to that point.
24 *
25 *   @param x           Point on X-axis
26 *   @param y           Point on Y-axis
27 *   @param z           Point on Z-axis
28 *   @return double[]  Angles and radius to a point in a sphere
29 *                   0: r
30 *                   1: a
31 *                   2: c
32 */
33 private double[] cartCorToSphere(double x, double y, double z){
34     double[] tmp = new double[3];
35
36     tmp[0] = sqrt(x*x + y*y + z*z);
37     tmp[1] = toDegrees(acos(z/tmp[0]));
38     tmp[2] = toDegrees(atan2(y, x));
39     return tmp;
40 }

```

Java code 5.2: Helping methods for finding the direction values of the tool vector.

```
1 /**
2  * Method to find length from center and radius of a Spherical ←
3  * cap
4  * @param r          Radius of sphere
5  * @param circleAngle Angle to point in sphere
6  * @return array      Array containg
7  *                   0: z    Distance from center
8  *                   1: a_r   Radius of new circle
9 */
10 private double[] findSphericalCap(double circleAngle, double r){
11     double[] tmp = new double[2];
12     double h;
13
14     tmp[0] = r*sin(toRadians(90-circleAngle));
15     h = r - tmp[0];
16     tmp[1] = sqrt(h*(2*r-h));
17     return tmp;
18 }
```

Java code 5.3: Method to find length from center and radius of a Spherical cap.

5.2 Using the G-code

To make sure the G-code that was generated from the Java program worked, the complete G-code was pasted into HSMWork's backplot function. With this function, it is possible to see which direction the tool vector will have at all times. It is also possible to simulate the print as a head/head CNC. Examples of the simulation in HSMWork's backplot function can be seen in Figure 5.5, 5.7 and 5.6. In the examples, the white lines are the path of the nozzle defined with X, Y, Z and the yellow lines are the direction of the tool vector defined by I, J, K. These simulations are a great way of checking the G-code visually before testing it on the 5-axis 3D printer.

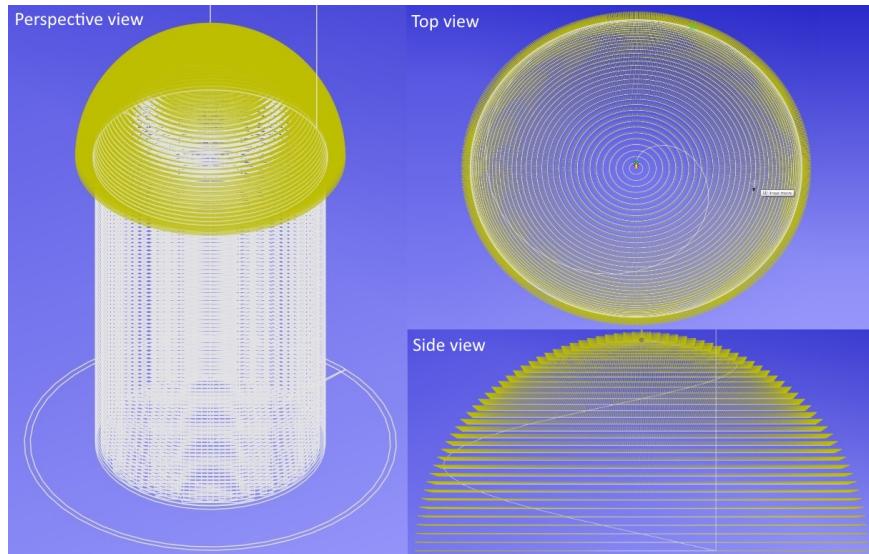


Figure 5.5: G-code simulation of surface smoothing test on xy-plane

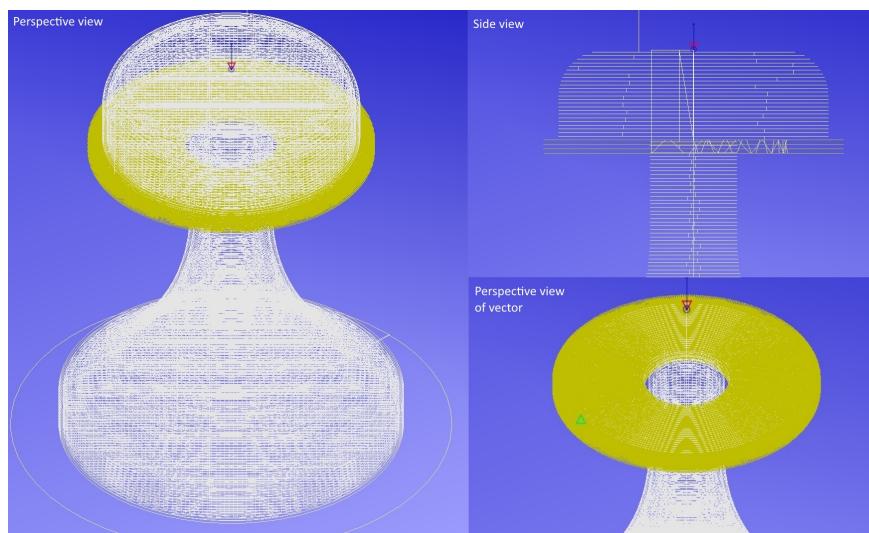


Figure 5.6: G-code simulation of support test

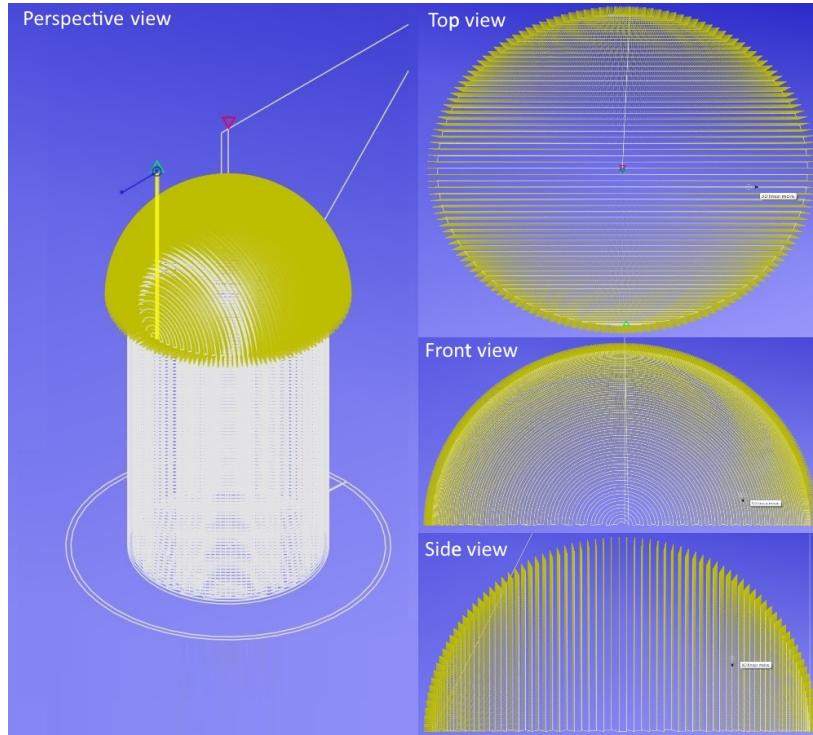


Figure 5.7: G-code simulation of surface smoothing test on xz-plane

5.2.1 Calibrating the printer

Before it is possible to print with the system the machine coordinate system (MCS) origin must be found and the A-axis calibrated. As mentioned in section 3.1.3 the MCS origin has to be where the rotary axes intersect. To find this origin, the height of the print plate have to be calibrated first. This is done by using a point with a known distance to the origin. In this case the top of the A-axis mount was used. And adjusting it with the calibration system created in section 3.3.2, and improved in section 3.8.2. In this case the top of the A-axis mount was used. Then the nozzle is moved to a point on the print plate that is around the zero point of the Y-axis. At this point the calibration screws of the print plate are moved up to the nozzle and adjusts it so that the nozzle at the given intersection point is barely touching the print plate. Then the same process is repeated for the other calibration screw by rotating the C-axis. This process is done until the C-axis can rotate 360° with the nozzle barely touching the print plate. When the print plate is calibrated and the Z-axis origin is known, the A-axis is calibrated. To do this the nozzle is moved to a point along the Y-axis, for example $Y = 30$, and then moved down till the nozzle barely touches the print plate, with the value of the Y-axis and the Z-axis offset, the A-axis offset can be found with eq. (5.5). Then the same procedure is done with the opposite side of the Y-axis, in this example $Y = -30$. This process is repeated until the nozzle can go between the two Y-axis points with the nozzle barely touching the print plate.

$$A_{Offset} = \arctan\left(\frac{Z}{Y}\right) \quad (5.5)$$

When the intersection point is found and the print plate is aligned to these points, it is possible to find the MCS origin of the printer by printing out plastic on an assumed zero point, then rotating the C-axis 90° three times and print plastic for each of these rotations. By measuring these points, we can find the X- and Y-axis offset and calibrate the printer with regard to these. This is repeated until it is possible to print out a point and rotate the C-axis 360° without moving around this point. When the origin in X, Y and Z are found these can be calibrated with the calibration function on the Ormerod.

The A-axis must be calibrated between each print since the A-axis may lose steps during a print, which mentioned in section 3.10 occurs on this system.

5.2.2 Printing with the G-code

When printing the parts Polylactic acid (PLA) were used as material, and the material was melted at 200° C. The base structures for the surface test and the top and bottom structures for the support test were printed with a layer height of 0.4 mm and sliced with Cura or Slic3r.

Surface test

As mentioned in section 5.1.1 two methods were created to test if the printer could create surfaces which are more smooth than a traditional FDM printer. These methods have input parameters of:

1. **resolution:**
The distance between each generated point in the G-code
2. **layerDistance:**
The distance between each plastic thread
3. **layerHeight:**
The height of the printed layer
4. **circleRadius:**
Radius of the sphere printed on
5. **platformHeight:**
The height up to the beginning of the 5-axis print
6. **layers:**
Amount of layers printed with the 5-axis system
7. **cOffset:**
Offset on C-axis to avoid same changing point in G-code (only for surface along X- and Y-axis)

The *circleRadius* and *platformHeight* was set based on the base structure of the tests. In this case *circleRadius* = 10 and *platformHeight* = 30. The resolution was set to 0.1 mm to have a high resolution and still have a G-code file of an acceptable size. The *cOffset* was set to 10°. Different values for *layerHeight*, *layerDistance* and *layers* were tested on the parts. It was found early on that the right value for *layerDistance*, and *layerHeight* between 0.1 – 0.2 mm could eliminate the need for extra layers on the surface. Since the resulting parts had an smooth enough surface after one layer. Two prints of each of the methods, with *layerHeight* = 0.2 mm, *layerDistance* = 0.5 mm and *layers* = 1, can be seen in Figure 5.8.



(a) Surface along X- and Y-axis. (b) Surface along X- and Z-axis.

Figure 5.8: Bad result of surface smoothing tests.

As it is possible to see from this picture, the layer distance between the FDM thread is too high. By changing the value of this to 0.4 mm and keeping the rest, the results improve as seen in Figure 5.9 and 5.10.



Figure 5.9: Result of surface smoothing test on xy-plane



Figure 5.10: Result of surface smoothing test on xz-plane

Support test

The method for generating G-code for a support structure for the 5-axis systems have input parameters of:

1. **resolution:**
The distance between each generated point in the G-code
2. **layerDistance:**
The distance between each plastic thread
3. **layerHeight:**
The height of the printed layer
4. **circleRadius:**
Radius of the cylinder printed on
5. **platformHeight:**
The height to the top of the beginning of the 5-axis print
6. **layersHeight:**
Amount of layers in height
7. **layersWidth:**
Amount of layers in width
8. **cOffset:**
Offset on C-axis to avoid same changing point in G-code (only for surface along X- and Y-axis)

The *circleRadius* and *layerDistance* was set based on the base and top structure of the test. In this case *circleRadius* = 5 and *layerDistance* = 0.5. The *layersHeight* was set so that *layerHeight***layersHeight* was equal to the radius of the top structure, which is 30 mm, minus the radius of the cylinder printed on, which is 10 mm. The *layersHeight* then becomes 25. The *layersWidth* was set to four, to get a support structure with a breath of $4 * \text{layerDistance}$ which are in this case 2 mm. On the first test the platform height was set to the height of the platform minus half the *layerDistance*, thus 39.75 mm. This value created a support structure which was underneath the top of the base, and therefore the top part was

not able to stick completely to the support structure. This can be seen in Figure 5.11. This happened because the first layer in the top G-code was not able to connect with the top of the support structure. To handle this the *platformHeight* was set to 40.1 mm, this way the top part managed to stick to the support part, as seen in Figure 5.12.



Figure 5.11: Bad result of support test



Figure 5.12: Result of support test

Part III

Conclusion and Results

Chapter 6

Results and Analysis

This chapter will present and analyze the modeling results of the final version. Then the printing results will be presented and analyzed.

6.1 Modeling results

The first version had some severe problems with the accuracy of the rotary axes. The angular error for the A- and C-axis was between 0.4° - -1.1° and as mentioned in section 3.5, this angular error was periodic. When comparing this periodic angular error with the motor revolution, the reason for the error was clear. As mentioned in section 3.5 this error existed because the worm was not connected to the motor axle properly, and therefore the worm had a elliptic movement and a periodic effect on the worm gear. In the first version there was no proper way to compensate for this error. Because this error, the system would not be able to print out test parts properly. It was necessary to create a new system where this fault and others were removed.

When final version of the 5-axis 3D printer was assembled, there were still some problems with finding the perfect distance between the worm and the worm gear, as mentioned in section 3.9. Because of this the A- and C- axis still had some backlash, (1° - 2°). In the A-axis the backlash was irrelevant since it was limited to 0° - 90° , and thereby preloaded because of the weight of the C-axis mount. On the C-axis, this can be a problem if the rotation direction changes. During the printing with the 5-axis system this problem was not severe enough to show, but it could be a problem when working with parts requiring high accuracy. Therefore, this issue should be addressed in future work on the system.

Testing the accuracy of the system, as done in section 3.10, shows that the system has an angular error that lies within 0.15° - -0.15° . These values are within the accuracy error of the digital inclinometer, which are $\pm 0.2^\circ$. Thereby it is difficult to draw a conclusion from these numbers. The rotating axes should be measured with more accurate measuring tool in future work.

In section 3.10 the rotary axes were stress tested to find out if they are rigid enough to be used for larger prints. In the tests it was shown that the C-axis did have any problems when stress tested. The A-axis had an angular error between 0.2° - 0.4° after a stress test. This angular error appears because the stepper motor driving the A-axis is losing small amounts of steps for each movement. This can be fixed by either reducing the speed of the A-axis or loosening the worm gear system for the A-axis.

6.2 Printing results

When creating G-code for the base and top structures both Cura and Slic3r were used to generate the G-code. When printing parts from these slicers it was clear that Cura made more efficient paths than Slic3r, and therefore used less time while printing these parts. Although Slic3r used longer time to print, there was a better final result. This is because Slic3r is the recommended slicer for the Ormerod, and there are profiles where the parameters for the print are adjusted to fit the Ormerod perfectly. It would be possible to do the same for Cura, but due to limited time was this not prioritized. Therefore, all the base and top structures for the finished parts shown in this section were sliced with Slic3r.

Surface test

As mentioned in section 1.2, one of the ways to get a better surface finish on a 3-axis FDM printer, is to reduce the layer height of a printed part. Reducing layer height will increase the number of layers in the print. This creates a longer traveling distance for the printer and thus a longer printing time. In Figure 6.2 it is possible to see that the print time for the same part roughly doubles when the layer height is halved.

As mentioned in section 5.1, two strategies for the surface test were used; paths along the X- and Y-axes, and paths along the X- and Z-axes. For simplicity these will be called XY-surface and XZ-surface.

To examine the surface results of the printed parts a microscope was used to see the layers with a minimum height of 0.1 mm, the position of the parts in the microscope can be seen in Figure 6.1. When examining these results it came apparent that the surface finish with the 5-axis system was an improvement compared to the surface the 3-axis FDM printer was able to create with a layer height of 0.1 mm, 0.2 mm and 0.4 mm. Comparing the XY-surface and XZ-surface to a part with 0.1 mm layer height at the side of the surface, it is possible to see a small improvement of the surface, as seen in Figure 6.3 and 6.7. When comparing the same parts at the top the improvement becomes more apparent, as seen in Figure 6.4 and 6.8. Comparing the XY-surface and XZ-surface to parts with 0.2 mm and 0.4 mm layer height, the improvement with the 5-axis system is clear, as can be seen in Figure 6.5, 6.6, 6.9 and 6.10.

When examining the XY-surface and XZ-surface with a microscope it is possible to see that the surfaces are still not perfect. There are different possible reasons for this:

- The resolution of the G-code is set to 0.1 mm, this makes the printer to go in straight lines, which is 0.1 mm around the the surface.
- The nozzle is created to drag out the plastic on a flat surface, and therefore it will create facets for each point it goes to in the 5-axis system.
- There could still be angular errors in the axes, which could not be measured with the digital inclinometer that was used.
- The 5-axis system is printing on a base structure which has a layer height of 0.4 mm. Since the surface layer is only 0.2 mm thick, it is possible that the surface of the base structure affects the surface of the 5-axis surface.

The XY-surface as seen in Figure 6.2, is the fastest of these strategies. This is because the Z-axis, which is the slowest axis on this system, moves for every point in the path on the XZ-surface. While on the XY-surface, the Z-axis only moves when changing circle. By using a system which has a faster Z-axis, the times should start to equalize.

Since the XZ-surface is limited by this system, the times from the XY-surface will be analyzed. As seen in Figure 6.2 the print time for the whole print with one XY-surface layer is 56 minutes. When comparing this with the print time for the base structure with 0.2 mm and 0.1 mm layer height, the print time with the XY-surface is reduced by 32.6% and 65.7% respectively. These numbers show that this surface method can dramatically decrease the print time and create a smoother surface than a print with layer height of 0.1 mm.



Figure 6.1: Position of parts when using microscope

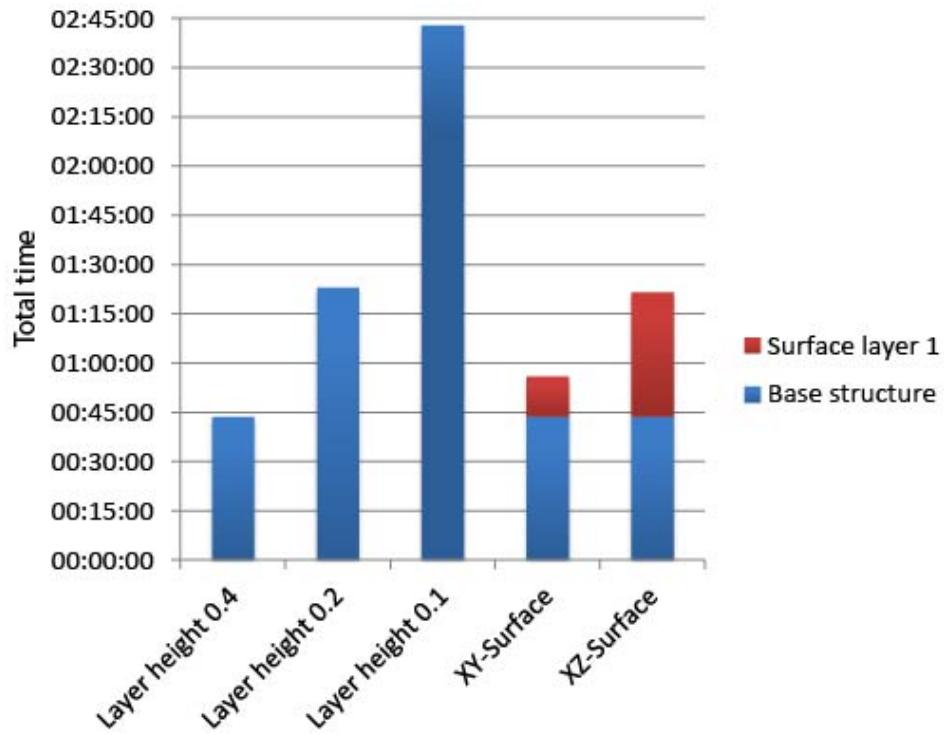


Figure 6.2: The print time for the surface tests

Figures 6.3 - 6.10 shows the magnified view comparison between XY- and XZ-surface and the base structure with 0.1 mm, 0.2 mm and 0.4 mm layer height. The picture to the left shows how the parts are aligned and the picture to the right is the magnified view. In the pictures are the XY- and XZ-surface always to the right and the base structure to the left.

In the following figures 6.3 - 6.6 XY-surface is shown:

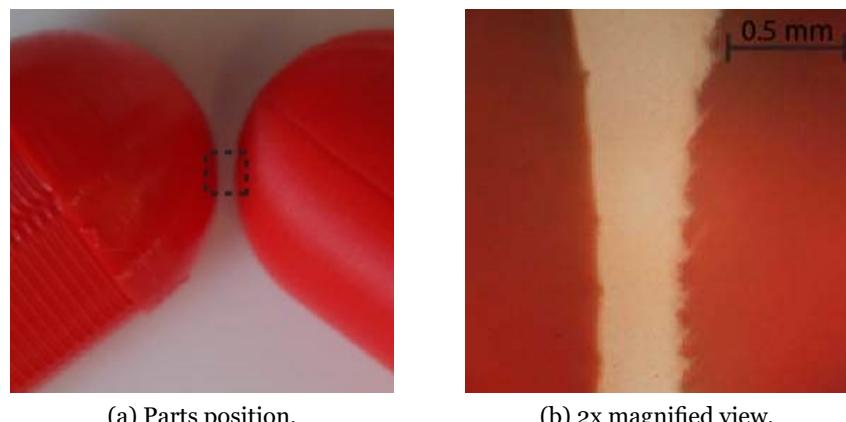
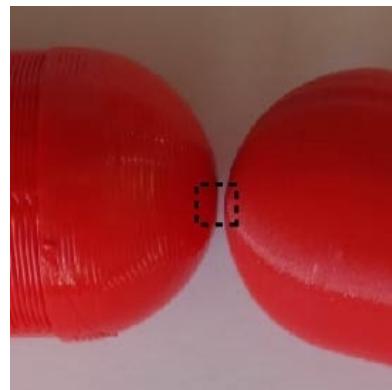
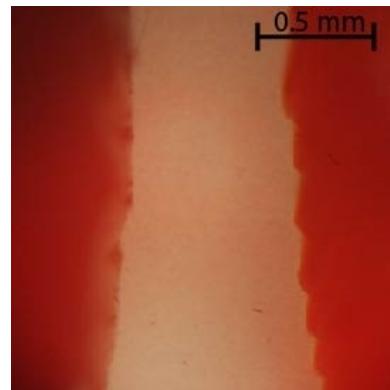


Figure 6.3: Part with layer height 0.1 mm and XY-surface.



(a) Parts position.

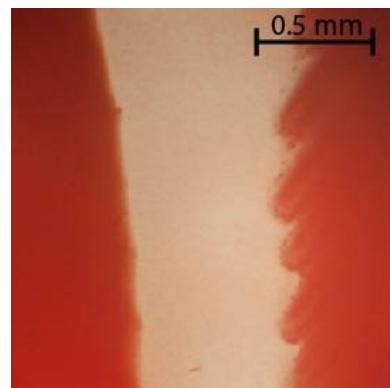


(b) 2x magnified view.

Figure 6.4: Part with layer height 0.1 mm and XY-surface.

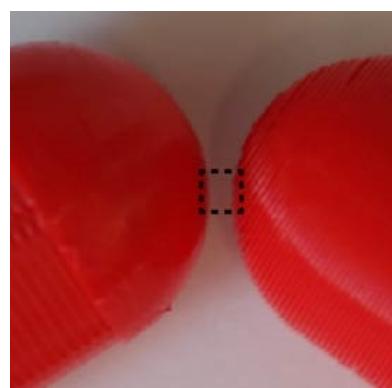


(a) Parts position.

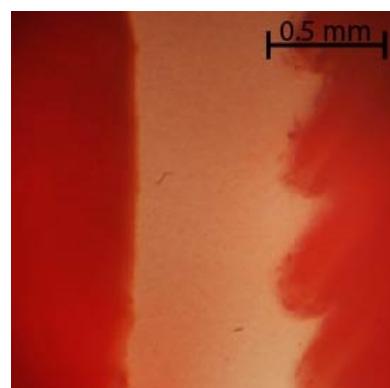


(b) 2x magnified view.

Figure 6.5: Part with layer height 0.2 mm and XY-surface.



(a) Parts position.



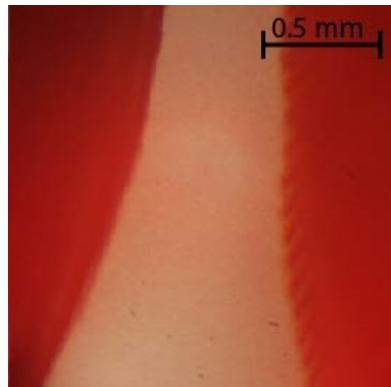
(b) 2x magnified view.

Figure 6.6: Part with layer height 0.4 mm and XY-surface.

In the following figures 6.7 - 6.10 XY-surface is shown:

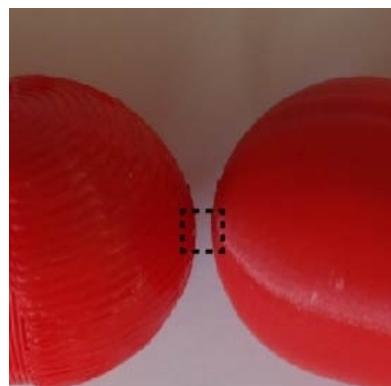


(a) Parts position.

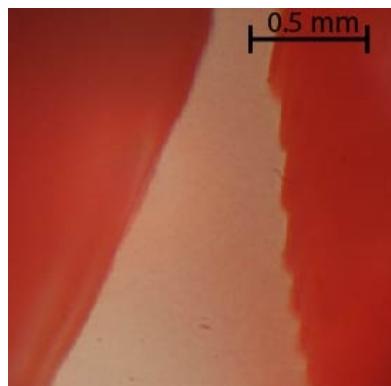


(b) 2x magnified view.

Figure 6.7: Part with layer height 0.1 mm and XY-surface.



(a) Parts position.

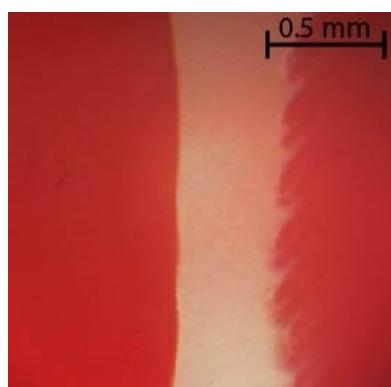


(b) 2x magnified view.

Figure 6.8: Part with layer height 0.1 mm and XZ-surface .



(a) Parts position.



(b) 2x magnified view.

Figure 6.9: Part with layer height 0.2 mm and XY-surface.

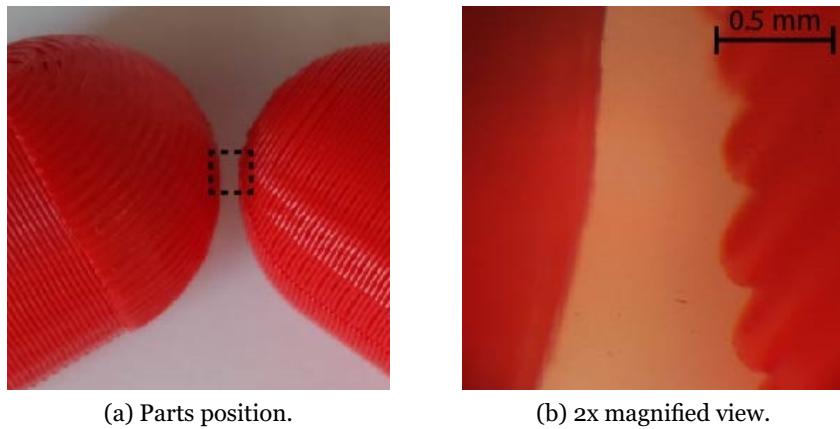


Figure 6.10: Part with layer height 0.4 mm and XY-surface.

Support test

As mentioned in section 1.2 another problem for FDM printers, is that they have problems with overhanging structure without using support. Most FDM printers will start to have problems with printing overhangs that are over 45° , and none of them are able to print a overhang that is 90° . To see if the 5-axis system could eliminate this limitation a part with a 90° overhang was printed. The part can be seen in Figure 5.12. For comparison the same part with support structure was printed out on the same system. In Figure 6.11 the part printed out on the 5-axis system can be seen side by side with a part that are printed out with support structure. The part printed without the 5-axis system needs a lot of support structure to print the 90° overhang. This support structure is both a hassle to remove and needs extra material to be created, 19% for this part.

In Figure 6.12 the time comparison between these two parts can be seen. As seen from this graph, the 5-axis system are 16.1% faster than the same structure with support. This is not a significantly decrease of the print time, however if the support structure (which takes 23 minutes) had an infill lower than 100% as it currently has, the print time will be reduced. For this multi step part there are internal layers which are completely filled, this is unnecessary and uses both extra time and material. By creating the G-code for the whole part instead of dividing it into three parts these layers would not be created. By doing this and reducing the infill of the support structure, the the printer should be able to reduce the print time and material used for the support structure significantly.

This shows that a 5-axis FDM printer can reduce both time costs and material cost for structures which need support material.



Figure 6.11: Comparison of part with support structure and same part printed on the 5-axis system. Part printed with 5-axis system to the left, and part with support structure to the right.

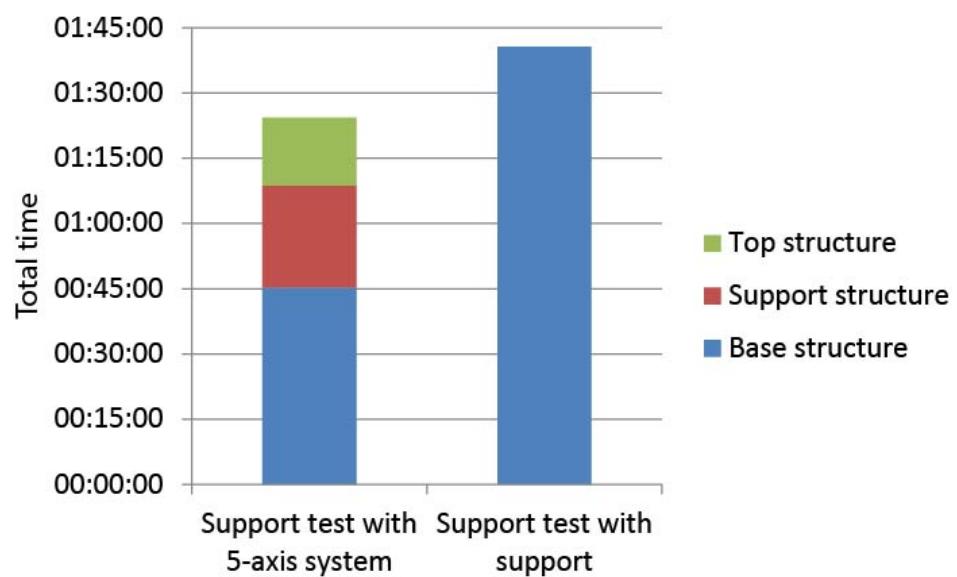


Figure 6.12: The print time for the support tests

Chapter 7

Discussion and Conclusion

7.1 General discussion

In this section there is a general discussion around the different aspects of the 5-axis system. For simplicity the first version will be called the first version and the second and final version will be called the Pentarod in this chapter.

7.1.1 The Pentarod

The designing process of the 5-axis system has been a big part of this thesis. This is because the rotary system had to be designed from the bottom and information about how this could be was not found during the research of this thesis, as mentioned in section 2.9. As mentioned in section 6.1 there were created two versions of the 5-axis system, due to severe problems with the first version. Fortunately the ideas from the first version could be transferred to the Pentarod, and therefore the designing process of this version were more straight forward than the first.

The Pentarod shows that it is possible to modify a 3-axis FDM printer to a 5-axis FDM printer. With access to a finished design, it is possible to easily modify most 3-axis FDM printers to work as a 5-axis system. All the parts that are needed can in most cases be printed out on the printer that is being modified, or bought in a hardware or electronic store.

As the second version and still a prototype, the Pentarod has a high accuracy. With the right calibration it has the potential to print out parts with an even higher accuracy. There are still some issues with the printer. The nozzle is still a limitation for the relation between the A-axis angle and the Z' distance between the print plate and the nozzle. And it is not possible to calibrate the rotary axes without user input. However it should be possible to program a calibration process into the hardware by using the sensor attached to the X-axis. And by creating a thinner and longer nozzle it should be possible to eliminate the limitation between the A-axis angle and the Z' distance between the print plate and the nozzle.

7.1.2 System compatibility

The G-code in this system uses the format of “G1 X Y Z I J K F E” as mentioned in Section 4.2. By using this G-code syntax, where it assumes that all the movement are done by the nozzle, the system does not depend on G-code which is created specific for this system. By doing this the Pentarod can be used to test slicers that are created to work with different systems. If other 5-axis FDM printers are developed with the same G-code syntax, these printers should be able to use the same G-code regardless of their configuration.

7.1.3 Test prints

The parts that have been printed out on the 5-axis system have shown that it is possible to remove some of the major limitations a 3-axis FDM printers have today. The parts still have a few problems; the amount of plastic printed, the distance between the layers, the difficulty of making the top part of the support test stick to the support structure, etc. However, all of these problems can be removed by adjusting the parameters for the printed parts and creating a complete G-code of the printed part.

In section 6.2 it was possible to see that the 5-axis system actually creates a more accurate surface than a 3-axis FDM printer can create with a really low layer height. And by finding better parameters this surface could, in theory, be even better. Another benefit with the 5-axis system for surfaces is the ability to control which way the tread of the surface is laying. Thus, it should be possible to create parts that could easily slide on each other.

7.1.4 Advantages with a 5-axis FDM system

As already mentioned in section 6.2, the 5-axis system is able to print both a better surface and parts without support in a shorter time than a 3-axis FDM printer. Since the 5-axis system removes the need of support, and therefore the material cost for printed parts. Another benefit with a 5-axis system is the possibility to control the direction a thread is being printed. This should make it possible to create structures which are stronger in multiple directions and create structures which can bend easily in multiple directions, or specified directions.

7.2 Conclusion

In section 1.3 this thesis asks if it is possible to eliminate the two main problems a 3-axis FDM printer has, by adding two more axes to the printer.

During this research, Pentarod, a 5-axis FDM printer has been designed, assembled, programed and tested. This has been done by extending an already existing 3-axis FDM printer with two more axes. In addition 5-axis G-code for different parts have been created, to test if a 5-axis system can eliminate the two main problems the FDM technology has.

The 5-axis system that has been created is still a prototype and there are still problems, as were mentioned in section 6.1. Despite these problem, this research have shown that it is possible to create a 5-axis FDM printer by adding a rotary system to a 3-axis FDM printer. It has also been proven that by rotating the printed part on a FDM printer, a 5-axis system can create better surface finish on a shorter time than a 3-axis FDM printer. And it can build 90° overhanging structures without the need of support.

With this it is possible to conclude that the project has achieved both of its goals in section 1.4, and thereby the project has been a success.

However, for the 5-axis FDM printer to be commercially available there is still a lot of research to be done. One of the main issues is the lack of a 5-axis slicer for 5-axis 3D printing. As mentioned in section 1.4, this project will be released as open source after this thesis has been delivered. Hopefully this will create an interest for 5-axis 3D printing in the open source movement, and grow interest in the development of 5-axis 3D printers and slicers.

While working on this project three videos were put on YouTube; a video¹ showing the XY-surface being printed, a video² showing the XZ-surface being printed, and a video³ showing the support test being printed. These videos were well received and 3ders.org⁴ and 3Dprint.com⁵ wrote articles about the 5-axis system after seeing these videos.

7.3 Future work

During the work on this thesis, it became apparent that the system still has room for improvements that should be addressed in further work.

One of the main improvements for the 5-axis system will be a longer and thinner nozzle. By adding this, the system will not be as limited as it is now. A longer and thinner nozzle would give the possibility to reach lower layers at a higher angle. In addition, the printer will be able to print places where the current nozzle would not be able to reach because of its size.

The A- and C- axes should also be tested with more accurate measuring tool. The system should be adjusted to make sure it is as accurate as possible. The backlash on the C-axis should also be addressed, this could be done by calibrating the distance between the worm and the worm gear, or a preload can be added to the C-axis.

¹YouTube video of XY-surface: <https://www.youtube.com/watch?v=7p7EHIVIJnE>

²YouTube video of surface test XZ: <https://www.youtube.com/watch?v=mqDrHYVOTVA>

³YouTube video of support test: https://www.youtube.com/watch?v=Y07vC_qE3sw

⁴Article on 3ders.org: <http://www.3ders.org/articles/20150704-an-amazing-open-source-5-axis-3d-printer-built-by-university.html>

⁵Article on 3DPrint.com: <http://3dprint.com/77400/5-axis-3d-printer/>

Due to time constrains, the printer was only tested to see if it was possible to print better surfaces and without support. For both of these different parameters, such as layer height, layers and speed should be researched. This to see how much impact they have on the finished result. Another case that also should be looked into is the possibility to create both flexible and stronger parts. This can be done by having layers that are crossing each other, and going in other directions than the ordinary 3-axis FDM printers does.

This thesis concludes that a 5-axis FDM printer can print structures with 90° and lower overhang without support. The test structure used for this test have a simple overhanging structure, and therefore it should be looked into what problems a 5-axis system can get when printing a more complex overhanging structure. It should also be considered if still adding some support to the structure, but from higher layers and from other angles than normally, can improve the finish and allow a higher complexity of the overhanging structure.

To create a system that is more user friendly, a auto calibration method for the rotary axes should also be added to the printer. This could be done by using the X-axis sensor, which would measure the distance between the nozzle and the print plate at different points. The system could then compute the angular error for the A-axis.

During this thesis one of the main shortages software wise, has been the lack of a 5-axis slicer for 3D printer. There are some development in this field. For example Bread⁶ by Nick Parker is in development. Bread can slice 3D models in 3D space, compared to ordinary slicers that slices in 2.5D. With Bread, Parker hopes to be able to remove some of the limitations ordinary 3D printers have today. For example the stair like structure on surfaces that are not parallel with the print plate. After a mail correspondence with Parker, he said that it should be easy to change the output of Bread to give tool angles in addition to ordinary G-code. This way it should be possible to use his slicer on a 5-axis 3D print system.

⁶Bread: <https://github.com/nick-parker/Bread>

Bibliography

- [1] *5axismaker Homepage*. 5axismaker. URL: <http://5axismaker.com/>.
- [2] *A Guide to the CNC Milling Machine*. CNC Machine HQ. URL: <http://cncmachinehq.com/guide-to-cnc-milling-machine/> (visited on 10/07/2015).
- [3] *About RepRap*. RepRap. 2014. URL: <http://www.reprap.org/wiki/About> (visited on 11/07/2015).
- [4] *Additive Manufacturing*. eos. URL: http://www.eos.info/additive_manufacturing/for_technology_interested (visited on 10/07/2015).
- [5] Mark Albert. *Vector Programming Eases Five-Axis Aerospace Machining*. 2012. URL: <http://www.mmsonline.com/articles/vector-programming-eases-five-axis-aerospace-machining> (visited on 15/01/2015).
- [6] Karlo Apro. *Secrets of 5-axis machining*. Industrial Press Inc., 2008.
- [7] George B. Arfken, Hans J. Weber and Frank E. Harris. *Mathematical Methods for Physicists*. Seventh Edition. Academic Press, 2013. ISBN: 978-0-12-384654-9. URL: <http://www.sciencedirect.com/science/article/pii/B9780123846549000037>.
- [8] V.S. Bagad. *Mechatronics*. Pune: Technical Publications, 2009. ISBN: 9788184314908.
- [9] Margherita Barile and Eric W Weisstein. *Arc*. 2014. URL: <http://mathworld.wolfram.com/Arc.html> (visited on 15/01/2015).
- [10] Chandi Campbell. *Cartesian, Delta, and Polar: The Most Common 3D Printers*. 2015. URL: <http://makezine.com/2015/03/10/cartesian-delta-polar-common-3d-printers/> (visited on 10/07/2015).
- [11] Wikimedia Commons. *Backlash*. File: Backlash.svg. 2010. URL: <https://commons.wikimedia.org/wiki/File:Backlash.svg>.
- [12] RepRap community. *Duet*. 2015. URL: <http://reprap.org/wiki/Duet> (visited on 30/06/2015).
- [13] *Comparison of computer-aided design editors*. URL: https://en.wikipedia.org/wiki/Comparison_of_computer-aided_design_editors (visited on 20/01/2015).
- [14] Mike Cope. *5-Axis Buzz Words: learn the lingo*. 2013. URL: <http://blog.hurco.com/blog/bid/263615/5-Axis-Buzz-Words-learn-the-lingo> (visited on 15/01/2015).

- [15] Mike Cope. *5-Axis Programming: programming with tool vectors*? 2013. URL: [http://blog.hurco.com/blog/bid/309807/5-axis-programming-programming-with-tool-vectors?source=Blog_Email_\[5-Axis%20Programming:%20\]](http://blog.hurco.com/blog/bid/309807/5-axis-programming-programming-with-tool-vectors?source=Blog_Email_[5-Axis%20Programming:%20]) (visited on 15/01/2015).
- [16] cplusplus.com. *function acos*. 2014. URL: <http://wwwcplusplus.com/reference/cmathacos/> (visited on 15/01/2015).
- [17] cplusplus.com. *function atan2*. 2013. URL: <http://wwwcplusplus.com/reference/cmathatan2/> (visited on 15/01/2015).
- [18] cplusplus.com. *function cos*. 2013. URL: <http://wwwcplusplus.com/reference/cmathcos/> (visited on 15/01/2015).
- [19] cplusplus.com. *function sin*. 2013. URL: <http://wwwcplusplus.com/reference/cmathsin/> (visited on 15/01/2015).
- [20] *Cura homepage*. Ultimaker. URL: <https://ultimaker.com/en/products/cura-software> (visited on 10/11/2014).
- [21] *Datasheet for 42BYGHM809*. Available at <http://cdn.sparkfun.com/datasheets/Robotics/42BYGHM809.PDF>. Sparkfun. 2011. (Visited on 15/03/2015).
- [22] *Datasheet for SM-42BYG011-25*. Available at <https://www.sparkfun.com/datasheets/Robotics/SM-42BYG011-25.pdf>. Sparkfun. 2010. (Visited on 15/03/2015).
- [23] Florent De Dinechin and Matei Istoan. ‘Hardware implementations of fixed-point Atan2’. In: () .
- [24] *Design Goals of the Java Programming Language*. Oracle. URL: <http://www.oracle.com/technetwork/java/intro-141325.html> (visited on 10/07/2015).
- [25] J. Dong. *Rapid Response Manufacturing: Contemporary methodologies, tools and technologies*. Manufacturing Systems Engineering Series. Springer US, 2012. ISBN: 9781461563655. URL: <https://books.google.no/books?id=FGTIBwAAQBAJ>.
- [26] Alan Earls and Vinod Baya. *The road ahead for 3-D printers*. pwc. URL: <http://www.pwc.com/us/en/technology-forecast/2014/3d-printing/features/future-3d-printing.jhtml>.
- [27] *Fortus 250mc information*. Stratasys. URL: <http://www.stratasys.com/3d-printers/design-series/fortus-250mc>.
- [28] *Fortus 250mc specification*. Stratasys. URL: http://usglobalimages.stratasys.com/Main/Secure/System_Spec_Sheets-SS/Fortus-Product-Specs/Fortus250mcSellSheet-US-ENG-09-13%20WEB.pdf?v=635457711364285862.
- [29] *Gear Backlash*. Neugart. 2001. URL: <http://www.neugartusa.com/Service/faq/Gear%20Backlash.pdf>.
- [30] Mats Høvin. *Coordinates systems for 5 axis milling*. 2014. URL: <http://heim.ifi.uio.no/matsh/inf4500/lec/0.php?s=7&l=14&d=0> (visited on 09/01/2015).

BIBLIOGRAPHY

- [31] Mats Høvin. *Lecture 10: CNC machine intro.* 2014. URL: <http://heim.ifi.uio.no/matsh/inf4500/lec/ind.php?l=10&d=0> (visited on 09/01/2015).
- [32] Mats Høvin. *Lecture 11: Milling objectives and Mechanical parts of a CNC machine.* 2014. URL: <http://heim.ifi.uio.no/matsh/inf4500/lec/ind.php?l=11&d=0> (visited on 09/01/2015).
- [33] Mats Høvin. *Lecture 12: Commonly used 3D operations and mechanical guides.* 2014. URL: <http://heim.ifi.uio.no/matsh/inf4500/lec/ind.php?l=12&d=0> (visited on 09/12/2014).
- [34] Mats Høvin. *Lecture 9: 3D printing (additive manufacturing) technologies.* 2014. URL: <http://heim.ifi.uio.no/matsh/inf4500/lec/ind.php?l=9&d=0> (visited on 09/01/2015).
- [35] *HSMWorks homepage.* HSMWorks. URL: <http://www.hsmworks.com/> (visited on 10/06/2015).
- [36] *Inclinometer purchase site.* Ebay. URL: <http://www.ebay.com/itm/Electronic-Digital-Inclinometer-Clinometer-Tilt-Angle-Meter-Slope-Gauge-Level-/141056327255?hash=item20d79cbe57> (visited on 06/02/2015).
- [37] *Information about JK42HS34-1334A.* Available at <http://www.jkongmotor.com/Product.asp?Action=View&ProductID=437&Catalog=13>, version JK42HS34-1334.jkongmotor. 2015. (Visited on 15/03/2015).
- [38] *Insight.* SMG3D. URL: http://www.smg3d.co.uk/3d_design_software/insight_software.
- [39] *Kickstarter for 5axismaker.* Kickstarter. 2014. URL: <https://www.kickstarter.com/projects/2003668803/5axismaker-first-ever-affordable-5axis-multi-fabri>/description (visited on 11/10/2014).
- [40] Karim R Lakhani and Eric von Hippel. ‘How open source software works: “free” user-to-user assistance’. In: *Research Policy* 32.6 (2003), pp. 923–943. ISSN: 0048-7333. DOI: [http://dx.doi.org/10.1016/S0048-7333\(02\)00095-1](http://dx.doi.org/10.1016/S0048-7333(02)00095-1). URL: <http://www.sciencedirect.com/science/article/pii/S0048733302000951>.
- [41] Hod Lipson and Melba Kurman. *Fabricated: the new world of 3D printing.* Indianapolis, IN: Wiley, 2013. ISBN: 9781118350638.
- [42] Tony Lock. *Duet - Arduino Due Compatible 3D Printer Electronics.* 2013. URL: <http://blog.think3dprint3d.com/2013/12/Duet-Arduino-Due-compatible-3DPrinter-controller.html> (visited on 10/02/2015).
- [43] Chu Anh My. ‘Integration of CAM Systems into Multi-axes Computerized Numerical Control Machines’. In: *Knowledge and Systems Engineering (KSE), 2010 Second International Conference on.* Oct. 2010, pp. 119–124. DOI: 10.1109/KSE.2010.30.
- [44] *Objet Connex 3D Printers.* Stratasys. URL: <http://www.stratasys.com/3d-printers/design-series/connex-systems>.

- [45] *Objet Connex 500 information at ROBIN wiki*. ROBIN. URL: <http://robin.wiki.ifi.uio.no/Connex500>.
- [46] *Objet Connex 500 Technical Specifications*. 3dvision. URL: <http://www.3dvision.com/pdf/3D-printing/design-series/Objet-Connex-500.pdf>.
- [47] *Printron homepage*. Printron. URL: <http://www.pronterface.com/> (visited on 10/11/2014).
- [48] RepRapPro. *Ormerod documentation*. 2014. URL: <https://reprapro.com/documentation/ormerod/> (visited on 15/11/2014).
- [49] Surjeet Sankararaj. *Worm Gear – Definition, Working and Applications*. 2013. URL: <http://mechteacher.com/worm-gear/> (visited on 10/07/2015).
- [50] SD3DPrinting. *FFF vs. SLA vs. SLS: 3D Printing*. 2013. URL: <http://sd3d.com/fff-vs-sla-vs-sls/> (visited on 25/07/2015).
- [51] *Slic3r homepage*. Slic3r. URL: <http://slic3r.org/> (visited on 10/11/2014).
- [52] *SolidWorks homepage*. SolidWorks. URL: <http://www.solidworks.com/> (visited on 10/06/2015).
- [53] M.W. Spong and S. Hutchinson. *Robot Modeling and Control*. Wiley, 2005. ISBN: 9780471649908. URL: <https://books.google.de/books?id=wGapQAAACAAJ>.
- [54] Collin Stoner. *Series and Parallel Motor Winding*. 2011. URL: <http://www.selene.co/Blog/2011/08/series-and-parallel-motor-winding/> (visited on 20/01/2015).
- [55] Bjarne Stroustrup. *Bjarne Stroustrup's FAQ*. URL: http://www.stroustrup.com/bs_faq.html#invention (visited on 10/07/2015).
- [56] Kamesh Tata et al. ‘Efficient slicing for layered manufacturing’. In: *Rapid Prototyping Journal* 4.4 (1998), pp. 151–167. DOI: 10.1108/13552549810239003. eprint: <http://dx.doi.org/10.1108/13552549810239003>. URL: <http://dx.doi.org/10.1108/13552549810239003>.
- [57] *Types of 3D printers or 3D printing technologies overview*. 3dprintingfromscratch. URL: <http://3dprintingfromscratch.com/common/types-of-3d-printers-or-3d-printing-technologies-overview/> (visited on 10/07/2015).
- [58] *Volcano - Print Faster, Stronger, Bigger*. E3D-ONLINE. 2014. URL: http://e3d-online.com/blog/volcano_release (visited on 20/07/2015).
- [59] Eric W. Weisstein. *Spherical Cap*. 2014. URL: <http://mathworld.wolfram.com/SphericalCap.html> (visited on 15/01/2015).
- [60] Eric W. Weisstein. *Spherical Coordinates*. 2014. URL: <http://mathworld.wolfram.com/SphericalCoordinates.html> (visited on 15/01/2015).

BIBLIOGRAPHY

- [61] *What is Java*. Oracle. URL: http://java.com/en/download/faq/whatis_java.xml (visited on 10/07/2015).
- [62] *Why 5-Axis?* Hurco. URL: <http://www.5-axis.org/index.php/why-5-axis-benefits-of-switching-to-a-5-axis-machine/> (visited on 10/01/2015).
- [63] *Worms and Worm Gears Information*. Globalspec. URL: http://www.globalspec.com/learnmore/motion_controls/power_transmission/gears/worms_worm_gears (visited on 10/07/2015).

Appendix A

G-code generator Code

```
1 import java.util.*;
2 import java.io.*;
3 import static java.lang.Math.*;
4
5 /**
6 * Generator for 5-axis g-code.
7 * All the methods are returning a LinkedList which are ←
8 * containing GcodeNode with all
9 * the information of a point in the g-code path.
10 * All the methods expect that the base of the part are ←
11 * centered in origin of the printer.
12 */
13 class Gcode_generator{
14     public double plasticPerkvadMM = 0.2078760;
15
16     /**
17      * Sphere along X- and Y-axis
18      * @param resolution      The distance between each ←
19      *                         generated point in the g-code
20      * @param layerDistance    The distance between each plastic ←
21      *                         thread
22      * @param layerHeight      The height of the printed layer
23      * @param circleRadius      Radius of the sphere printed on
24      * @param platformHeight   The height up to the beginning of ←
25      *                         the 5-axis print
26      * @param layers           Amount of layers printed with the 5←
27      *                         axis system
28      * @param cOffset          Offset on C-axis to avoid same ←
29      *                         changing point in g-code
30      * @return LinkedList       LinkedList containing the g-code ←
31      *                         nodes
32 */
33     public LinkedList<GcodeNode> sphereXY(double resolution, ←
34         double layerDistance, double layerHeight, double ←
35         circleRadius, double platformHeight, int layers, double ←
36         cOffset){
37         LinkedList<GcodeNode> tmpList = new LinkedList<GcodeNode>();
38
39         // Angles for A-axis and C-axis
40         double a, c = 0.0;
```

```

30     double layer = 1, sphereRadius;
31     double aResolution;
32     boolean dirUp = false;
33     double[] sphericalCap;
34
35     a = dirUp ? 90.0 : 0.0;
36
37     while(layer <= layers){
38         sphereRadius = (circleRadius + (layerHeight*layer));
39
40         // Using arc length to find the resolution along the A←
41         // axis
42         aResolution = (180*layerDistance)/(PI*sphereRadius);
43
44         if(a == 0.0) a += aResolution;
45
46         while(dirUp ? a >= 0 : a <= 90){
47             sphericalCap = findSphericalCap(a, sphereRadius);
48
49             tmpList.addAll(gcodeXYCircle(resolution, layerHeight←
50                 , sphericalCap[1], sphericalCap[0], c, ←
51                 platformHeight));
52             c += cOffset;
53
54             if(c > 360) c = c - 360;
55
56             a = dirUp ? a - aResolution : a + aResolution;
57         }
58
59         a = dirUp ? a + aResolution : a - aResolution;
60         layer++;
61         dirUp = !dirUp;
62     }
63
64     /**
65      * Sphere along X- and Z-axis
66      * @param resolution      The distance between each ←
67      *                         generated point in the g-code
68      * @param layerDistance    The distance between each plastic ←
69      *                         thread
70      * @param layerHeight      The height of the printed layer
71      * @param circleRadius     Radius of the sphere printed on
72      * @param platformHeight   The height up to the beginning of ←
73      *                         the 5-axis print
74      * @param layers           Amount of layers printed with the 5←
75      *                         axis system
76      * @return LinkedList       LinkedList containing the g-code ←
77      *                         nodes
78     */
79     public LinkedList<GcodeNode> sphereXZ(double resolution, ←
80         double layerDistance, double layerHeight, double ←
81         circleRadius, double platformHeight, int layers){
82         LinkedList<GcodeNode> tmpList = new LinkedList<GcodeNode>←
83             ();
84
85         double cResolution, sphereRadius;
86         double[] sphericalCap;

```

APPENDIX A. G-CODE GENERATOR CODE

```

88     int layer = 1, circleCounter = 1;
89     boolean xDirection = true, yDirection= true;
90
91     while(layer <= layers){
92         sphereRadius = circleRadius+layerHeight*layer;
93
94         // Using arc length to find the resolution along the C-axis
95         cResolution= (180*layerDistance)/(PI*sphereRadius);
96
97         while(yDirection ? cResolution*circleCounter < 180 : cResolution*circleCounter > 0){
98             sphericalCap = findSphericalCap(cResolution*circleCounter, sphereRadius);
99
100            tmpList.addAll(gcodeXZCircle(resolution, sphericalCap[1], sphericalCap[0], platformHeight, layerHeight, xDirection));
101
102            xDirection = !xDirection;
103            circleCounter = yDirection ? circleCounter+1 : circleCounter-1;
104        }
105        layer++;
106        yDirection = !yDirection;
107    }
108    return tmpList;
109 }
110
111 /**
112 * Circles along X- and Y-axis
113 * @param resolution      The distance between each generated point in the g-code
114 * @param layerDistance   The distance between each plastic thread
115 * @param layerHeight     The height of the printed layer
116 * @param circleRadius    Radius of the cylinder printed on
117 * @param platformHeight The height to the top of the beginning of the 5-axis print
118 * @param layersHeight   Amount of layers in height
119 * @param layersWidth    Amount of layers in width
120 * @param cOffset         Offset on C-axis to avoid same changing point in g-code
121 * @return LinkedList      LinkedList containing the g-code nodes
122 */
123 public LinkedList<GcodeNode> circleXY(double resolution, double layerDistance, double layerHeight, double circleRadius, double platformHeight, int layersHeight, int layersWidth, double cOffset){
124     LinkedList<GcodeNode> tmpList = new LinkedList<GcodeNode>();
125
126     double z, r;
127     int layerHeightCounter = 1;
128     int layerBreadthCounter = 0;
129     double c = 0.0;
130     boolean direction = true;
131
132     while(layerHeightCounter <= layersHeight && layerBreadthCounter <= layersWidth){
133         if(direction)
134             r = circleRadius + (c - cOffset) * layerHeight;
135         else
136             r = circleRadius + (c + cOffset) * layerHeight;
137
138         z = platformHeight - (layerHeightCounter * layerHeight);
139
140         GcodeNode node = new GcodeNode("G1 X" + r + " Y" + z + " F" + resolution);
141         tmpList.add(node);
142
143         if(direction)
144             c += 2 * PI * layerBreadthCounter * layerHeight;
145         else
146             c -= 2 * PI * layerBreadthCounter * layerHeight;
147
148         if(c >= 360 || c <= -360)
149             direction = !direction;
150
151         layerBreadthCounter++;
152     }
153
154     return tmpList;
155 }
```

```

122     z = platformHeight;
123
124     while(layerHeightCounter <= layersHeight){
125         r = circleRadius + layerHeight*layerHeightCounter;
126         layerBreadthCounter = direction ? 0 : layersWidth-1;
127
128         while(direction ? layerBreadthCounter < layersWidth : ←
129             layerBreadthCounter >= 0){
130             z = platformHeight-layerDistance*layerBreadthCounter←
131                 ;
132
133             tmpList.addAll(gcodeXYCircle(resolution, layerHeight←
134                 , r, 0.0, c, z));
135
136             layerBreadthCounter = direction ? ←
137                 layerBreadthCounter + 1 : layerBreadthCounter - ←
138                 1;
139
140             c += cOffset;
141             if(c > 360)
142                 c -= 360;
143         }
144         direction = !direction;
145         layerHeightCounter++;
146     }
147
148     /**
149      * Method to find length from center and radius of a ←
150      * Spherical cap
151      *
152      * @param r          Radius of sphere
153      * @param circleAngle Angle to point in sphere
154      * @return array      Array containing
155      *                   0: z      Distance from center
156      *                   1: a_r    Radius of new circle
157      */
158     private double[] findSphericalCap(double circleAngle, double ←
159                                         r){
160         double[] tmp = new double[2];
161         double h;
162
163         tmp[0] = r*sin(toRadians(90-circleAngle));
164         h = r - tmp[0];
165         tmp[1] = sqrt(h*(2*r-h));
166         return tmp;
167     }
168
169     /**
170      * Method to create g-code for circles along the X- and Y- ←
171      * axis.
172      *
173      * @param resolution      The distance between each ←
174      *                       generated point in the g-code
175      * @param layerHeight     The layer height of the print
176      * @param radius          Radius of the circle
177      * @param z               Height of the circle in the sphere

```

APPENDIX A. G-CODE GENERATOR CODE

```

171     * @param zOffset           Height of the platform the circle ←
172     * are printed on
173     * @param cStart            Offset along the C-axis to avoid same←
174     * changing point in the g-code
175     * @return LinkedList        LinkedList containing the g-code ←
176     * nodes
177     */
178     private LinkedList<GcodeNode> gcodeXYCircle(double resolution←
179             , double layerHeight, double radius, double z, double ←
180             zOffset, double cStart){
181         LinkedList<GcodeNode> tmpList = new LinkedList<GcodeNode>←
182             ();
183         GcodeNode tmpGN;
184
185         double x, y, c = cStart;
186         Double e = null;
187         double[] tmp;
188         double cRes = (180*resolution)/(PI*radius);
189         boolean first = true;
190
191         while(c <= 360+cStart){
192             x = radius * sin(toRadians(c));
193             y = radius * cos(toRadians(c));
194
195             tmp = cartCorToSphere(x, y, z);
196             tmp = cartAngleToVector(tmp[1], tmp[2]);
197
198             // Do not extrude to the first point, since this is a ←
199             // rapid move.
200             if(first) first = false;
201             else e = resolution * layerHeight * plasticPerkvadMM;
202
203             tmpGN = new GcodeNode(x, y, z+zOffset, tmp[0], tmp[1], ←
204                     tmp[2], e);
205
206             tmpList.add(tmpGN);
207             c += cRes;
208         }
209
210         return tmpList;
211     }
212
213     /**
214     * Method to create g-code for half circles along the X- and ←
215     * Z- axis.
216     *
217     * @param resolution          The distance between each ←
218     * generated point in the g-code
219     * @param layerHeight         The layer height of the print
220     * @param radius              Radius of the circle
221     * @param y                   Distance between sphere origin and ←
222     * this circle
223     * @param zOffset             Height of the platform the circle ←
224     * are printed on
225     * @param direction           Which direction this circle are going
226     * @return LinkedList          LinkedList containing the g-code ←
227     * nodes
228     */

```

```

216     private LinkedList<GcodeNode> gcodeXZCircle(double resolution←
217         , double layerHeight, double radius, double y, double ←
218         zOffset, boolean direction){
219         LinkedList<GcodeNode> tmpList = new LinkedList<↔
220             GcodeNode>();
221         GcodeNode tmpGN;
222
223         int counter = 0;
224         double circleCounter, aResolution, z, x;
225         Double e = null;
226         double[] tmp;
227         boolean first = true;
228
229         // Using arc length to find the resolution along the A↔
230         // axis
231         aResolution = (180*resolution)/(PI*radius);
232         circleCounter = 180/aResolution;
233
234         counter = direction ? 0 : (int)circleCounter;
235
236         while(direction ? aResolution*counter < 180 : ←
237             aResolution*counter > 0){
238             z = sin(toRadians(aResolution*counter)) * radius;
239             x = cos(toRadians(aResolution*counter)) * radius;
240
241             counter = direction ? counter+1 : counter-1;
242
243             tmp = cartCorToSphere(x, y, z);
244             tmp = cartAngleToVector(tmp[1], tmp[2]);
245
246             // Do not extrude to the first point, since this is ←
247             // a rapid move.
248             if(first) first = false;
249             else e = resolution * layerHeight * plasticPerkavadMM↔
250                 ;
251
252             tmpGN = new GcodeNode(x, y, z+zOffset, tmp[0], tmp↔
253                 [1], tmp[2], e);
254
255             // Do not extrude to the first point, since this is ←
256             // a rapid move.
257             if(first)
258                 first = false;
259             else
260                 tmpGN.e = resolution * layerHeight * ←
261                     plasticPerkavadMM;
262
263             tmpList.add(tmpGN);
264
265         }
266         return tmpList;
267     }
268
269     /**
270      * Method to find vector values for a point in a sphere based
271      * on the spherical angles to that point.
272      *
273      * @param a          A-axis angle
274      * @param c          C-axis angle

```

```
265     * @return double[] Vector values :
266     *
267     *          0: I
268     *          1: J
269     *          2: K
270     */
271     private double[] cartAngleToVector(double a, double c){
272         double[] tmp = new double[3];
273
274         tmp[0] = cos(toRadians(c))*sin(toRadians(a));
275         tmp[1] = sin(toRadians(c))*sin(toRadians(a));
276         tmp[2] = cos(toRadians(a));
277
278         return tmp;
279     }
280
281 /**
282 * Method to find angles and radius to a point in a sphere ←
283 * based
284 * on the Cartesian coordinates to that point.
285 *
286 * @param x           Point on X-axis
287 * @param y           Point on Y-axis
288 * @param z           Point on Z-axis
289 * @return double[]   Angles and radius to a point in a sphere
290 *          0: r
291 *          1: a
292 *          2: c
293 */
294     private double[] cartCorToSphere(double x, double y, double z←
295                                     ){
296         double[] tmp = new double[3];
297
298         tmp[0] = sqrt(x*x + y*y + z*z);
299         tmp[1] = toDegrees(acos(z/tmp[0]));
300         tmp[2] = toDegrees(atan2(y, x));
301
302         return tmp;
303     }
304 }
305
306 /**
307 * Node class for storing and creating string of generated g-←
308 * code
309 */
310 class GcodeNode{
311     public Double x, y, z, i, j, k, e, f;
312
313     public GcodeNode(Double x, Double y, Double z, Double i, ←
314                     Double j, Double k, Double e){
315         this.x = x;
316         this.y = y;
317         this.z = z;
318         this.i = i;
319         this.j = j;
320         this.k = k;
321         this.e = e;
322         this.f = null;
323     }
324
325     public String toString(){
326         String tmp = "G1";
327
328         return tmp;
329     }
330 }
```

```
320     if(x != null) tmp += String.format(" X%.3f", x);
321     if(y != null) tmp += String.format(" Y%.3f", y);
322     if(z != null) tmp += String.format(" Z%.3f", z);
323     if(i != null) tmp += String.format(" I%.5f", i);
324     if(j != null) tmp += String.format(" J%.5f", j);
325     if(k != null) tmp += String.format(" K%.5f", k);
326     if(f != null) tmp += String.format(" F%.5f", f);
327     if(e != null) tmp += String.format(" E%.5f", e);
328
329     return tmp;
330 }
331 }
332 }
```
