UNIVERSITY OF BATH

DEPARTMENT OF MECHANICAL ENGINEERING

Continuously-variable material properties in RepRap 3D printing

Submitted by Pia Taubert

for the MSc Dissertation Module ME50185

Oktober 2012

Copyright

Cheating and Plagiarism

"I certify that I have read and understood the section in the MSc Student Handbook on Cheating and Plagiarism and that all material in this assignment is my own work, except where I have indicated with appropriate references."

Name: Pia Taubert

Signed Date

Abstract

Additive manufacturing processes improve continuously so that it is now possible to create multi-coloured objects. Some machines using fused filament fabrication are capable of creating parts of dual colour, which is achieved with the use of separate extrusion nozzles for each colour. By actuating the nozzles alternately, dual-coloured objects are produced. This process however is limited since it would be necessary to use a very high amount of filament inputs to achieve multi-coloured objects which is not a feasible option. This research project is based on the idea to design an extrusion head capable of processing multiple filament inputs and also using these to achieve material of additional colours or mechanical properties by mixing the filament available.

An extrusion head design was developed that is capable of mixing two polylactic acid filaments of different colour or different mechanical properties. The extrusion head is to be used on a RepRap machine, which is an open-source additive manufacturing machine developed at the University of Bath. Modified and newly written software programmes were used in conjunction with an existing RepRap machine, which were adapted to meet the requirements imposed by the new extrusion head design. The resulting extrusion head set up in combination with the relevant software were tested by printing parts using several PLA filament inputs which were mixed in the extrusion head. Parts with fixed mixing ratios as well as certain colour transitions were printed to evaluate the mixing process, which led to the conclusion that it is possible to achieve thorough mixing, resulting in the expected printed parts. Similar transition prints were done using filament inputs of different mechanical properties. This showed that it is possible to obtain printed objects with gradually changing material properties.

This research was conducted at the University of Bath, United Kingdom, under the supervision of Dr A. Bowyer, Dr P. Iravani (both University of Bath, United Kingdom), Prof. T. Vietor and Mr H. Prüß (both Technische Universität Braunschweig, Germany).

Acknowledgements

I would like to thank Dr A. Bowyer for his relentless ideas and support which were a great help throught the entire project, Dr P. Iravani, Dr S. A. Macgregor and Mr R. Jones for their help and advice with many different aspects of this project, Prof. T. Vietor and Mr H. Prüß for supervising my project at my German University, Mr T. A. Dale for helping me solve issues that occured in the course of this project, and C. for making this entire project possible.

Contents

List of Figures

List of Tables

1 Introduction

RepRap is an open-source self-replicating 3D printer which uses fused-filament fabrication to build three-dimensional engineering components from a variety of thermoplastics but primarily polylactic acid (PLA) and acrylonitrile butadiene styrene (ABS). The current state of development is such that extruders used with RepRap machines are capable of processing a single filament input of uniform colour and uniform mechanical properties. The aim is to develop, test and use a new extrusion head for the RepRap machine that is designed to mix two or more polymers and to build objects with the result. The initial aim is to perform the mixing using the one polymer of different colours to evaluate the thoroughness of the mixing process. The next step covers the mixing of polymers of different mechanical properties. Since RepRap primarily processes PLA and ABS, and since PLA filament exists in hard material as well as soft and bendable material, the resulting design will be evaluated by mixing PLA filament of different colours as well as hard and soft PLA filament.

Additive manufacturing techniques become more and more popular and at the same time evolve more and more. Most additive manufacturing machines are only capable of producing objects of one single or two colours. Only one commercially available additive manufacturing machine using a certain kind of additive manufacturing process is capable of creating full-coloured prints similar to two-dimensional colour printers which will be discussed in chapter 3.4. Developing an extrusion head capable of achieving similar results as well as using materials of different mechanical properties makes it possible to use this development in combination with various applications. Parts of continuously-variable mechanical properties could be used to replace assemblies consisting of objects which have different mechanical properties, such as is the case with hinges. Using a part of continuously-variable mechanical properties would result in seamless objects. This function would come in use for applications where increased friction due to fluid flow has to be avoided. For example airfoil design, where control surfaces could be incorporated into the wing but can still be moved individually.

2 Aims and Objectives

The aim of this project was to develop, test and use a new extrusion head for RepRap machines that is designed to mix multiple polymers and to build three-dimensional objects with the resulting polymer mixture. Initially, one polymer in different colours will be used for the mixing process to simplify the evaluation process of the results. This will allow to easily evaluate the thoroughness of mixing. The next stage covers mixing polymers with different mechanical properties. This allows building objects with continuously varying mechanical properties.

During the course of the project, the new extrusion head will be used on an existing, fully assembled RepRap machine to limit the design process to features relevant for the extrusion and mixing process.

3 Literature review

3.1 Additive Manufacturing

Additive manufacturing (AM) is based on the principle of creating a physical three-dimensional part directly from a three-dimensional computer aided-design (CAD) model in an additive manner by joining material. The typical additive manufacturing process involves several steps whose complexity and more detailed elements depend on the chosen AM process and the complexity of the three-dimensional object. The first step consists of creating a CAD model which then is converted to a file format which is accepted by the additive manufacturing machine. This typically is the STL file format which describes the external closed surface of the original CAD model. The STL-file is then transferred to the machine which in turn is set up in order to prepare the additive manufacturing process. The physical part is then built, removed from the machine once the manufacturing process has finished, and post processed if required so that the part is ready to be used with the intended application. As the individual steps indicate, process planning is not required when creating parts using additive manufacturing. While other manufacturing processes, such as subtractive manufacturing, require an in-depth analysis of the part geometry to determine the order and requirements of individual machining steps, additive manufacturing simply requires knowledge of the basic part geometry, the type of AM process and the materials used. Additive manufacturing usually requires a single building step regardless of the part complexity, whereas a higher part complexity results in an increased amount of often iterative machining stages when using other manufacturing processes.

Current additive manufacturing processes create parts by adding material in layers where each layer represents a thin cross-section of the part which is derived from its CAD model. The finite thickness of a layer imposes the resolution of the final product which is only an approximation of the original CAD model. The effect of using different layer thicknesses on the part resolution compared to the CAD model are indicated in figure 3.1.1.

Additive manufacturing machines using different additive manufacturing processes mainly differ in the materials that can be used and the methods used to create and bond layers of material. This determines the part accuracy, the achievable material and mechanical prop-

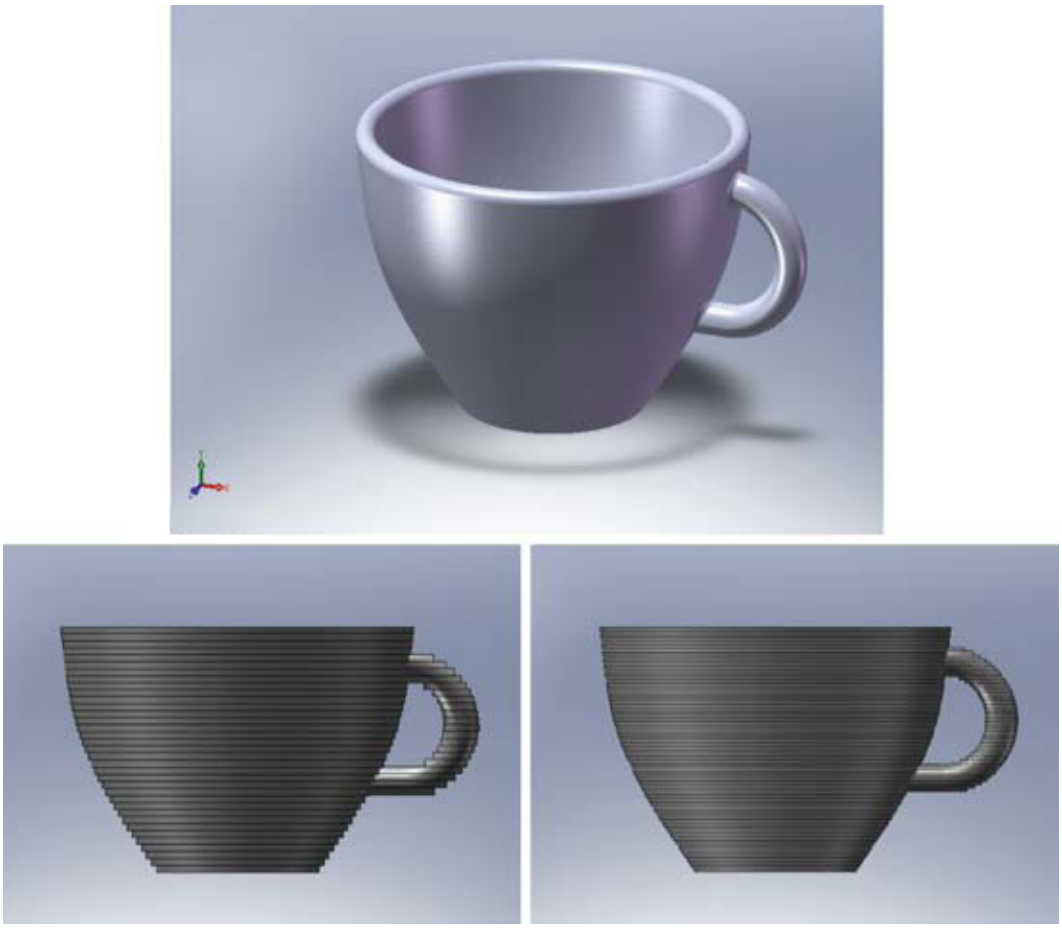


Figure 3.1.1: CAD image of a teacup with further images showing the effects of using different layer thicknesses [1]

erties, the speed of the AM process, the amount of post processing required and the overall cost of the AM machine and the AM process. Common additive manufacturing processes use different types of raw input material such as liquid polymers, discrete particles, molten materials and solid sheets for which a selection of the corresponding machining processes will be covered further.

3.1.1 Photopolymerisation

Systems based on the use of liquid polymers use photopolymerisation processes to cure radiation curable resins or photopolymers.

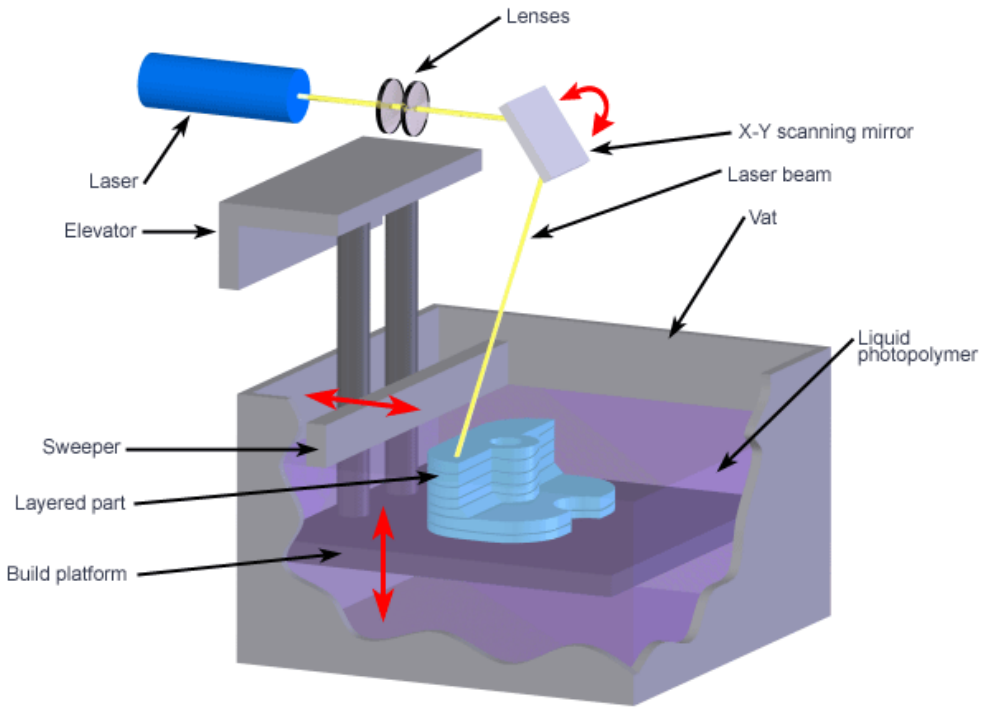


Figure 3.1.2: Working principle of Stereolithography [2]

These systems use a technology called stereolithography which is based on the principle that liquid polymers become solid when reacting with radiation of certain wavelengths, such as ultraviolet or visible light, which causes a chemical reaction. A basic machine setup for an AM machine that utilises this principle is displayed in figure 3.1.2. The three-dimensional object is built onto a build platform which is located in a vat filled with the liquid polymer. The laser beam starts localised, controlled chemical reactions which cause the surface of the liquid polymer to solidify locally. Once this process is finished for one layer of the object, the build platform moves down by a fraction so that the partly-built object can be covered with a new resin layer and the process can be repeated. Different machines based on this setup use different amounts and types of lasers, which allow curing material either point or layer wise. Machines using two individual lasers do not require the use of an elevator and the application of a new resin layer since the liquid polymer solidifies where the two laser

beams intersect which allows building the part below the surface of the liquid photopolymer. A very small layer thickness can be achieved by using stereolithography, hence parts of very high accuracy can be created.

3.1.2 Selective Laser Sintering

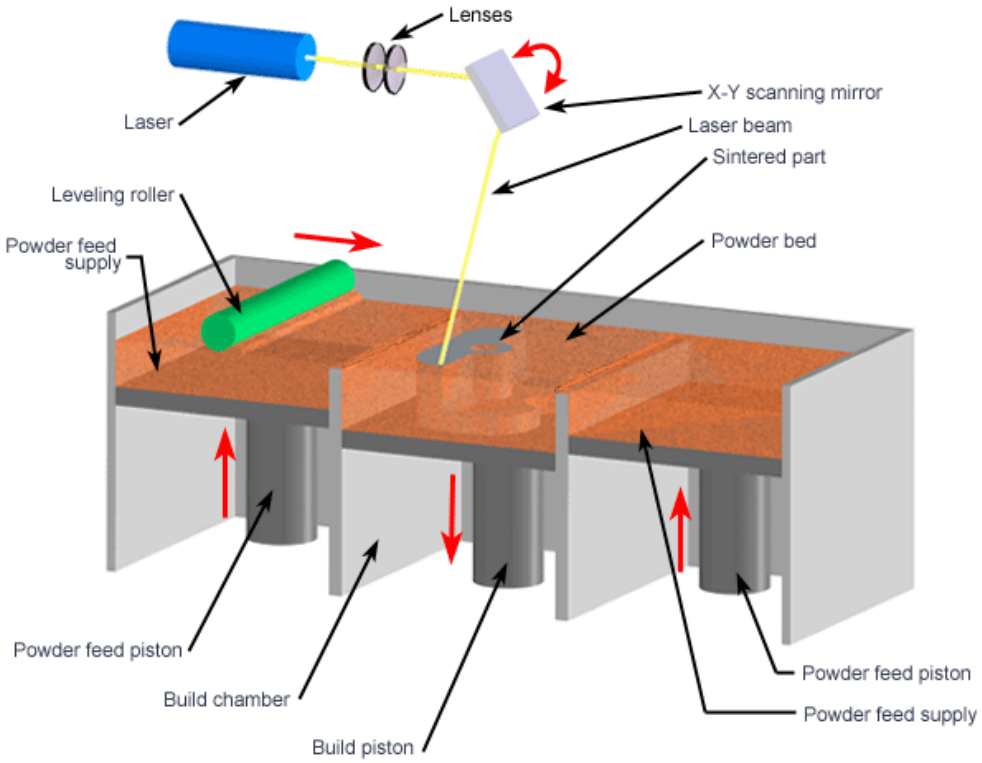


Figure 3.1.3: Working principle of Selective Laser Sintering [2]

A different additive manufacturing process is based on the process of powder bed fusion and forms three-dimensional parts by joining discrete particles that exhibit a thermoplastic behaviour. Machines that employ this process typically feature a build chamber, a powder feed supply mechanism and thermal sources. A layer of powder is applied to the print bed in the build chamber using a levelling roller to distribute the powder evenly. A thermal source, such as a laser, is then used to locally fuse powder particles within the powder layer which represents a cross-section of the three-dimensional part. When the sintering process has been finished for one layer, the print platform adjusts in height to allow for the next layer of powder to be added and the sintering process is repeated. The first commercialised

additive manufacturing process based on powder bed fusion was Selective Laser Sintering (SLS) whose basic working principle is displayed in figure 3.1.3 which is also representative for other powder bed fusion processes. Powder bed fusion processes do not require the use of support material, as the unsintered powder particles of previous layers act as a sufficient support structure. Furthermore, these processes can be used with a variety of materials such as polymers, metals, ceramics and composites which allows the resulting parts to be used for a variety of applications.

3.1.3 Three-Dimensional Printing

An additive manufacturing process similar to SLS is used by the Three-dimensional printing (3DP) process which also utilises powder bed fusion.

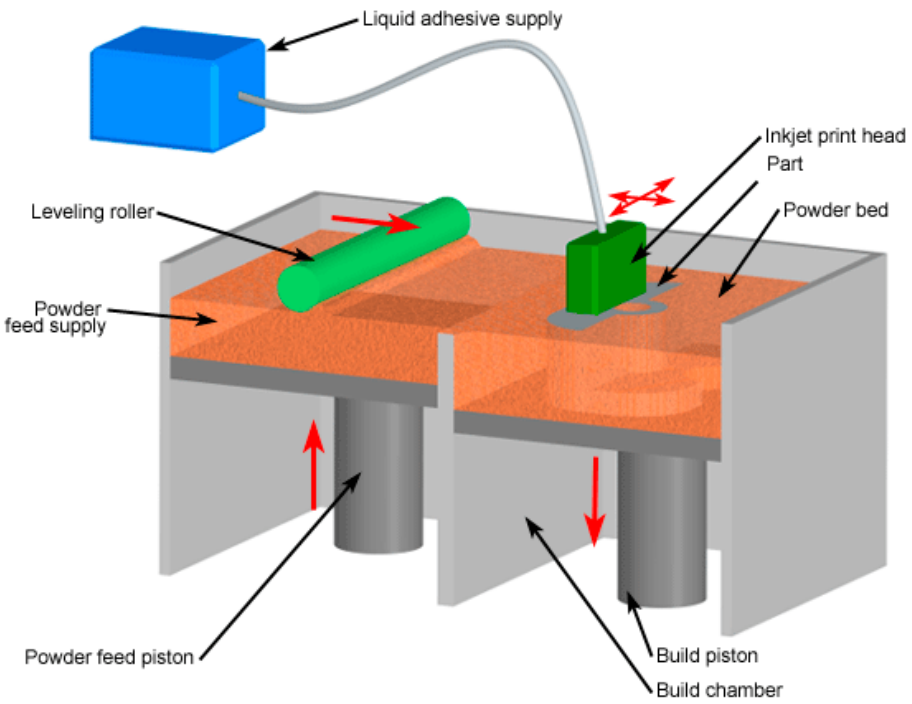


Figure 3.1.4: Working principle of Three-Dimensional Printing [2]

Similar to SLS machines, AM machines based on the 3DP technology feature a build chamber and a powder feed supply mechanism whereas 3DP machines feature a liquid adhesive supply and an inkjet print head instead of thermal sources. The print bed in the build chamber is

covered with a layer of powder which is supplied by a powder feed supply and distributed evenly by a levelling roller. Powder particles are then joined locally by depositing a liquid adhesive so that a cross-section of the build object is created. The build platform adjusts in height once the printing process has been completed for one layer and the entire process is repeated until the entire object is created. This AM process, whose general working principle is displayed in figure 3.1.4, allows the deposition of colour particles along with the liquid adhesive, hence enablingthe creation of multi-coloured objects. The loose powder particles act as support material so that no further support structure is required. Figure 3.1.4 shows the general working principle of Three-dimensional printing.

3.1.4 Fused Filament Fabrication

Fused filament fabrication (FFF), which is also known as Fused Deposition Modelling (FDM), is an additive manufacturing process that creates three-dimensional objects from extruded thermoplastics.

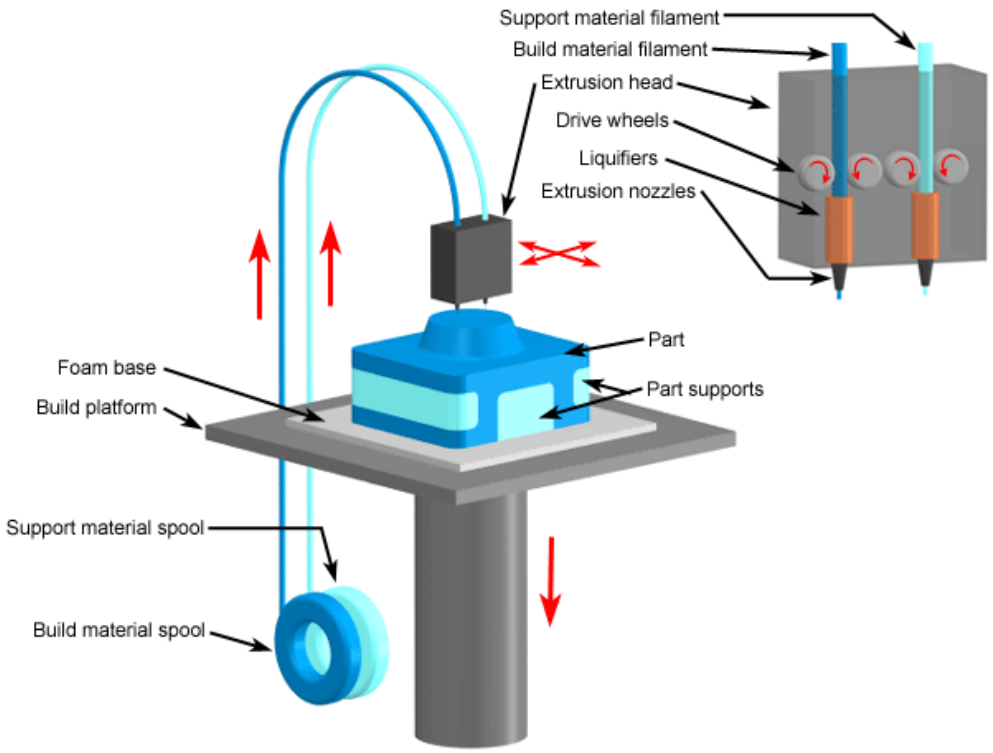


Figure 3.1.5: Working principle of Fused Filament Fabrication [2]

AM machines that utilise this process typically feature a build platform and an extrusion head that is capable of processing build material which is typically supplied in form of a spool of filament. Drive wheels in the extrusion head feed the filament into liquefiers where the material is heated up to a temperature just above its melting point. The material can then move through the nozzle easily which allows a controlled material deposition over the horizontal build plane so that a layer of the build object or a support structure can be built. The thermoplastic bonds to the previous layer and hardens immediately after being extruded from the nozzle. Either the build platform or the extrusion head adjust in height after one layer of the build part has been completed so that the next layer can be deposited. A typical machine setup featuring this process is shown in figure 3.1.5. Such processes usually process thermoplastics such as acrylonitrile butadiene styrene (ABS), polyamide (PA), polycarbonate PC, polyethylene (PE), polypropylene (PP) and polylactic acid (PLA).

3.1.5 Laminated Object Manufacturing

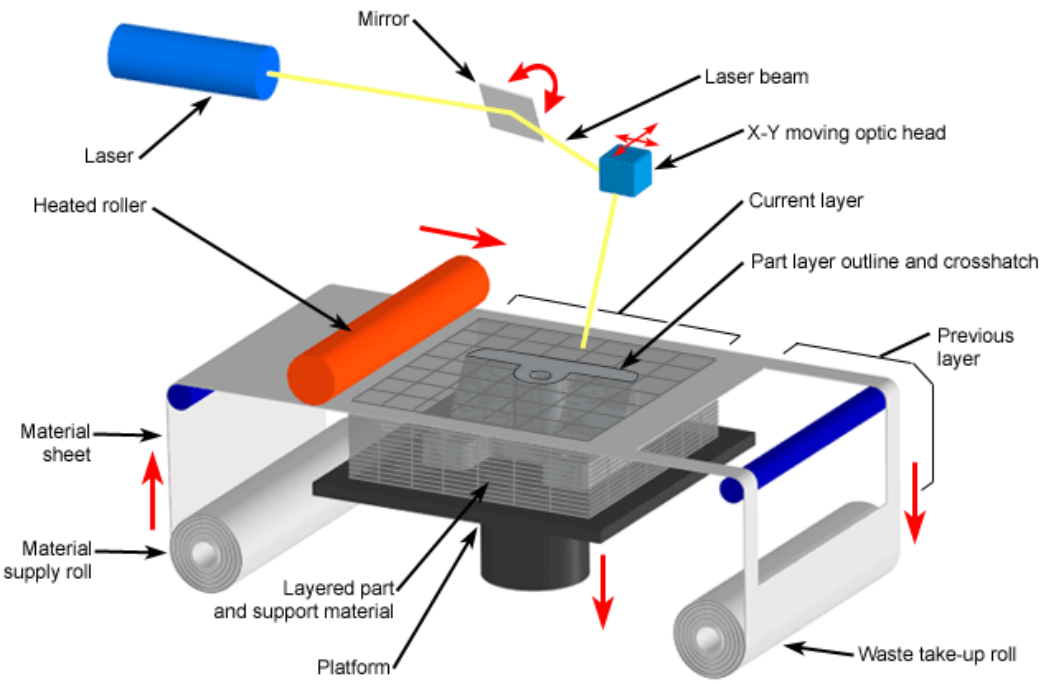


Figure 3.1.6: Working principle of Laminated Object Manufacturing [2]

A further technique used in additive manufacturing is Laminated Object Manufacturing (LOM) and is based on a sheet lamination process. An AM machine using this process features a material feed mechanism, a build platform, a heated roller and a laser. The material, which is supplied as adhesive-coated sheets, is placed on the build platform and bonds with the previous layer of material as pressure is applied by the heated roller. A laser then cuts the outline of the build part cross-section into the material sheet which is repeated until the entire part has been created using this entire process. The material which is not part of the built object acts as support material and can be removed after the manufacturing process has been completed. The working principle of this manufacturing process is displayed in figure 3.1.6. Initially, parts built by using a LOM process were created from paper material sheets that were joined using an adhesive bonding. Other processes have since been developed that utilise different build materials, cutting techniques and bonding methods.

3.2 Replicating Prototyper RepRap

RepRap, which stands for Replicating Rapid Prototyper, is an open-source desktop 3D printer that uses fused filament fabrication to create objects from thermopolymers such as polylactic acid (PLA) and acrylonitrile butadiene styrene (ABS). The RepRap project was initiated in 2004 by Dr Adrian Bowyer at the University of Bath and has since resulted in the development of several rapid prototypers. One of RepRap's key features is its capability to partly reproduce itself since it consists of a significant amount of parts that can be created using an additive manufacturing process. The remaining components that are part of a RepRap machine are easily and cheaply available standard mechanical and electronic parts which make a RepRap machine significantly cheaper than equivalent commercial machines and thus affordable for individuals. Due to its open-source nature, RepRap developers are located all over the world and every individual interested in this technology has the opportunity to contribute and develop it further. One of the existing RepRap 3D printer versions is shown in figure 3.2.1.

RepRap machines representing the current state of development feature one single extruder which is capable of processing one single strand of filament. Furthermore, widely available software used to control a RepRap machine is only capable of potentially dealing with mul-

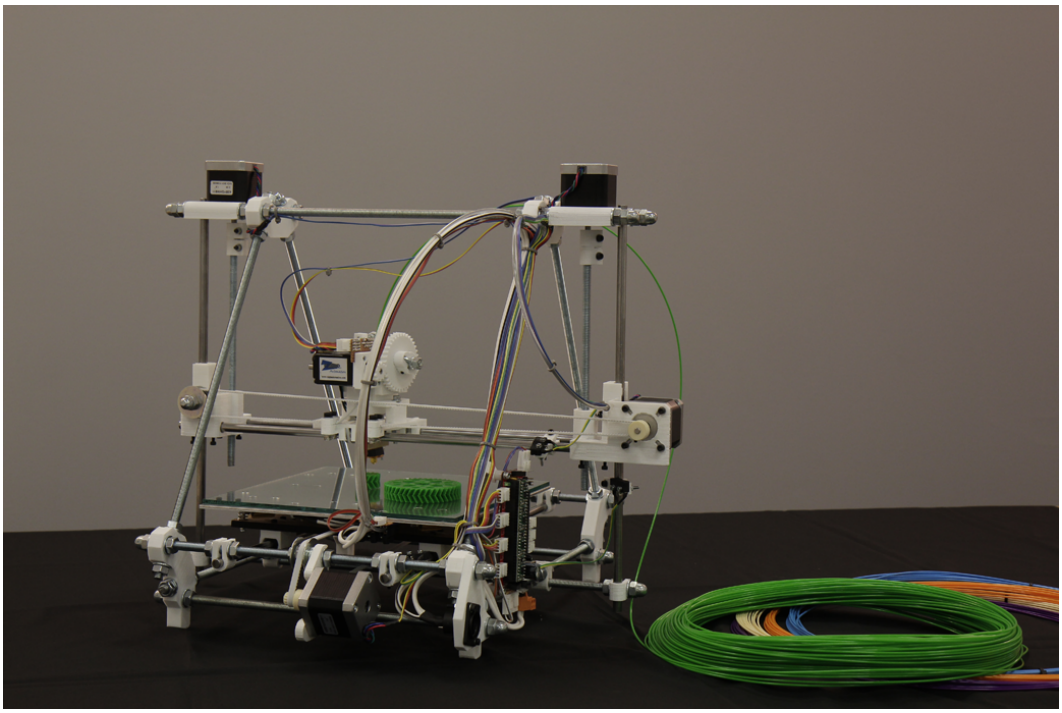


Figure 3.2.1: Replicating Rapid Prototyper RepRap

tiple extruder heads that are used alternately but it is not capable of processing information regarding multiple strands of filament that are driven simultaneously. A similar development state exists regarding the programming language that is used to interact with and control a RepRap machine which is G-code. Existing universal commands allow switching between different extruders of a RepRap machine but do not allow using different extruders simultaneously.

3.2.1 RepRap Manufacturing Process

A combination of different programmes is used to manufacture a three-dimensional object from its CAD model on a RepRap machine. These programmes are either commercially available programmes or open-source programmes that have been written for the use with RepRap machines. Each of the programmes performs a different task in the machining process. This covers converting the CAD model into the STL file format, converting this file into its respective G-code and controlling the RepRap machine so that the object can be printed from its G-code.

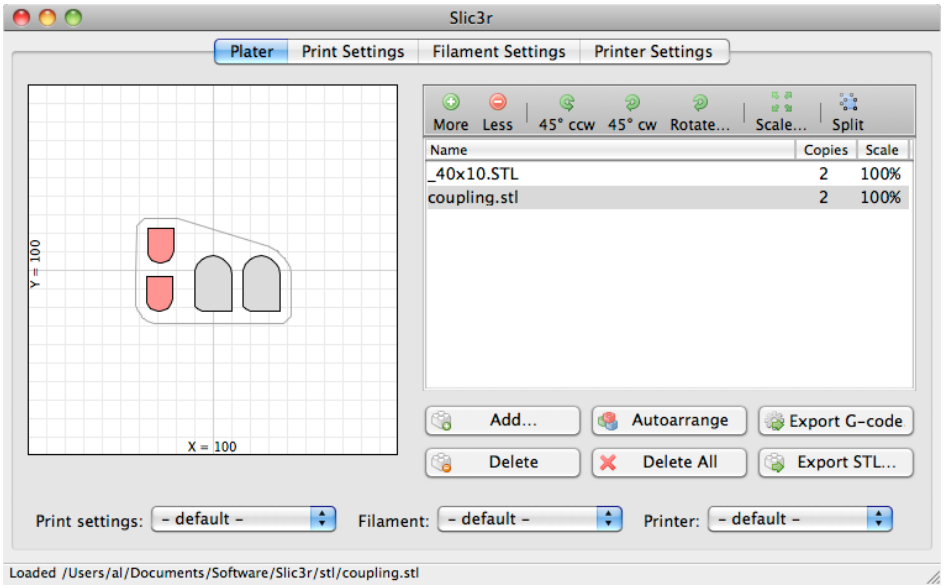


Figure 3.2.2: Slic3r User Interface [3]

To start with, an existing CAD model of the desired object has to be available in the STL file format which can be easily achieved by a variety of different CAD software. This STL file then has to be converted into a format that can be processed by a RepRap machine which is the programming language G-code. G-code is a computer numerical control programming language used by many computer controlled subtractive machining mechanisms. To create a G-code from an STL file, the STL model is sliced in horizontal layers of equal height and each of the layers is converted into G-code commands that are required to print the respective layer. The commands for each layer are then combined which results in the G-code command sequence that creates the entire object. Several programmes exist that are capable of converting the STL file of a CAD model into G-code that is compatible with software used to run a RepRap machine. One of these programmes, whose user interface is displayed in figure 3.2.2, is the open-source programme *Slic3r* which has specifically been created to be used with RepRap machines.

A combination of different programmes and electronics is required to control RepRap machines and process G-code commands. These can be chosen from a variety of options which have been designed and developed in order to be used with RepRap machines. One of the programmes required is a desktop programme which is the interface between a RepRap machine and a computer and performs the task of sending G-code commands to the RepRap

machine. One existing programme capable of performing this task is the open source programme *Printrun* whose user interface is displayed in figure 3.2.3.

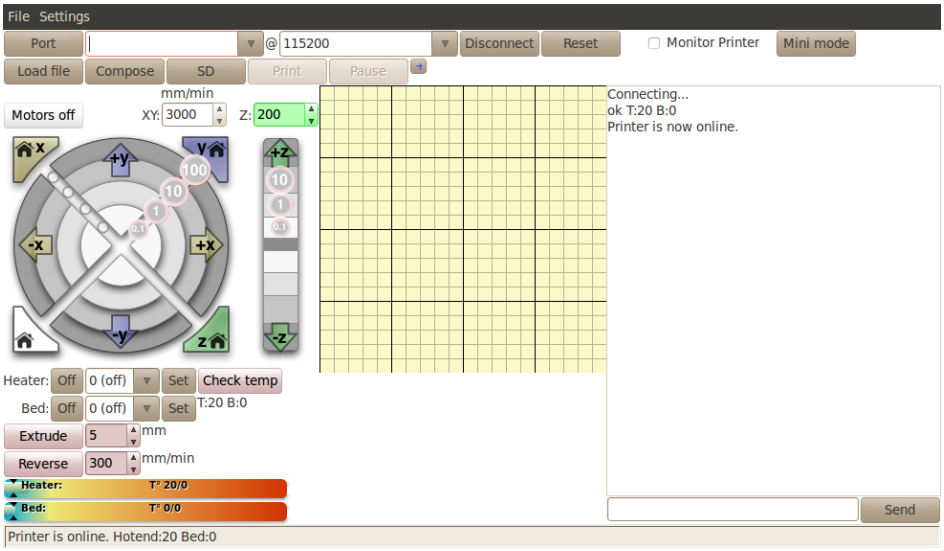
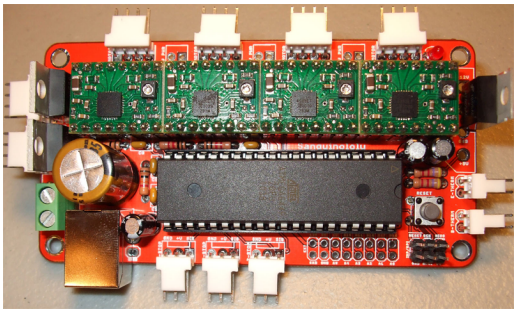


Figure 3.2.3: Printrun User Interface [4]

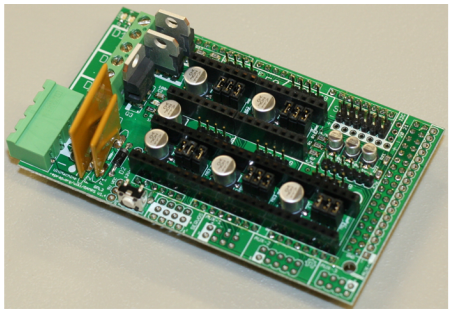
The G-code commands then have to be processed such that the RepRap machine reacts with the appropriate action. This task is performed by an additional programme, the RepRap firmware, which is uploaded onto a microchip that is part of the RepRap electronics. Again, different programmes exist that are capable of performing this task and each of these programmes has different advantages and disadvantages. The only advantage or disadvantage that will be mentioned at this point is the programme's potential capability of handling multiple extruder drive mechanisms. To this point, the open-source programme *Marlin* is the only well-established programme offering the possibility of using multiple extruders. This is not a feature typically used since the vast majority of RepRap machines are equipped with one single extruder.

Two different examples for potential RepRap electronics are shown in figure 3.2.4. The displayed *Sanguinololu* electronics are an all-in-one electronics board that is equipped with a microcontroller and four stepper motor drivers which allows to control four different stepper motors with this electronics set. The shown *Ramps 1.4* electronics do not contain a microcontroller so that this set of electronics has to be used in combination with a microcontroller based board such as the Arduino Mega. The *Ramps 1.4* electronics can be used

with five stepper motor drivers which offer the possibility to drive five different motors off the *Ramps 1.4*.



(a) Sanguinololu Electronics



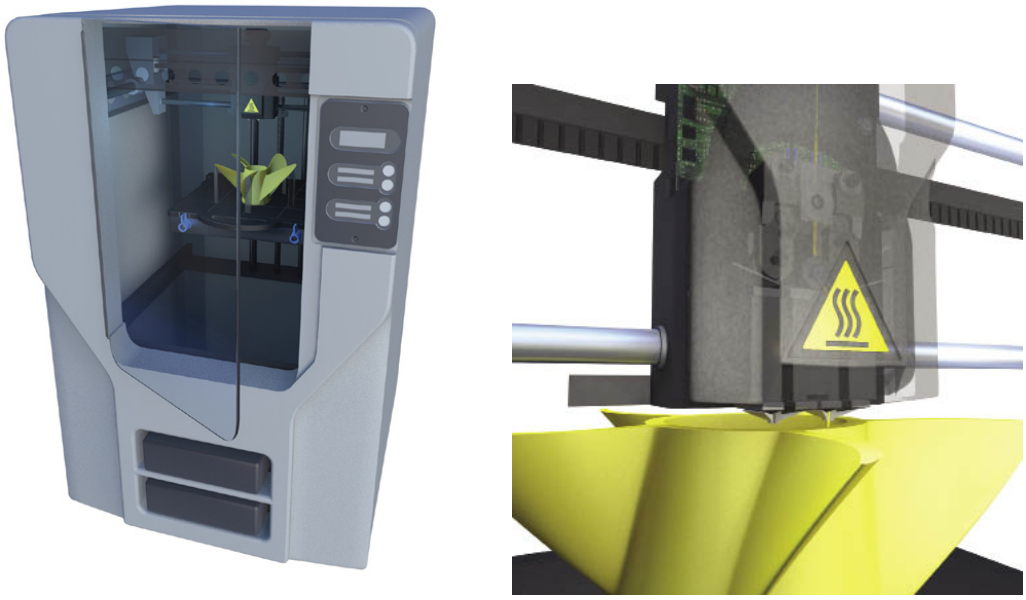
(b) Ramps 1.4 Electronics

Figure 3.2.4: Different RepRap electronics [4]

Each of the three axes as well as each extruder drive mechanism on a RepRap machine is driven by one motor each so that the *Sanguinololu* electronics are capable of controlling each of the three axes motors as well as one extruder stepper motor whereas the *Ramps 1.4* electronics are capable of controlling an additional second extruder motor.

3.2.2 Existing Fused Filament Fabrication Extruder Designs

Additive manufacturing machines based on fused filament fabrication are equipped with different extruder designs which all contain a drive mechanism and at least one extrusion nozzle attached to a liquefier. The main difference between existing fused filament fabrication extruder designs is the relative location of these two main components. Drive mechanism and extrusion nozzle can either both be located on the extruder carriage or the extrusion nozzle is attached to the extruder carriage while the drive mechanism is located on the frame.



(a) Stratasys 3D Printer

(b) Close-up of Stratasys 3D Printer extrusion head

Figure 3.2.5: Stratasys 3D Printer using direct filament drive [5]

If drive mechanism and extrusion nozzle are both located on the extruder carriage, the drive mechanism is generally located directly above the extrusion liquefier attached to the extrusion nozzle which offers very good support for the guided filament so that it is very unlikely to buckle or flex during this set up. Drive mechanism and extrusion nozzle both being located on the extruder carriage results in a relatively high volume and mass of components attached to the extruder carriage which typically is a disadvantage. This set up can hardly be used for using a multitude of filament inputs since the space on the extruder carriage is limited. Typical designs using this set up use a maximum of two filament inputs and thus extruder drive mechanisms simultaneously.

Locating the extruder drive mechanism on the machine frame imposes the necessity to use an extruder design capable of guiding the filament from the drive mechanism to the extrusion nozzle in order to reduce the filament's likeliness to buckle or flex. This can be achieved by using a Bowden cable set up which uses a flexible polymer tube to connect the extruder drive mechanism and the liquefier. This set up has the advantage of reducing volume and mass on the extruder carriage since only liquefier and extrusion nozzle are attached to the carriage. Therefore multiple filament inputs can be achieved with this set up. The use of a flexible polymer tube however increases the likeliness of the driven filament to buckle or flex slightly within the tube which results in non-consistent prints and is a slight disadvantage.

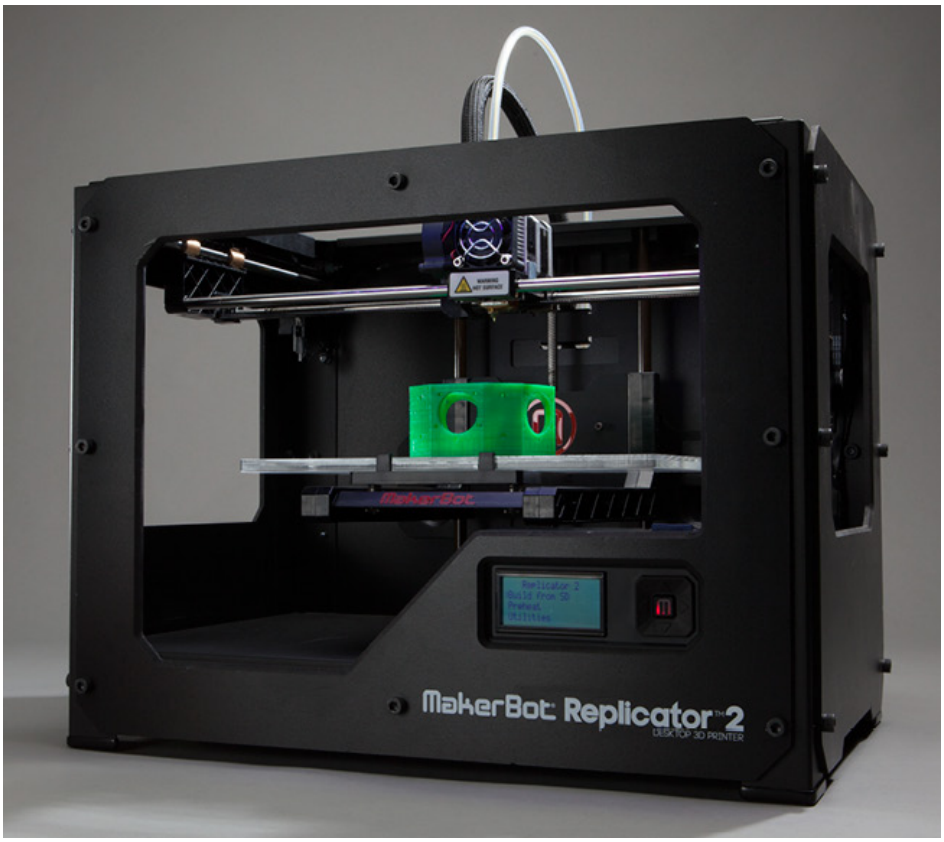


Figure 3.2.6: Makerbot 3D Printer using direct filament drive [6]

3D printers released by *Stratasys* and *Makerbot* feature different versions of direct filament drives as part of the extrusion system. Machines released by *Stratasys* use servo drives and lead screws as part of the extrusion head to drive filament which is indicated in figure 3.2.5 [5].

A different type of machine released by *Makerbot* on the other hand uses stepper motors as part of their machine's extrusion head to drive filament. This set up is shown in figure 3.2.6.

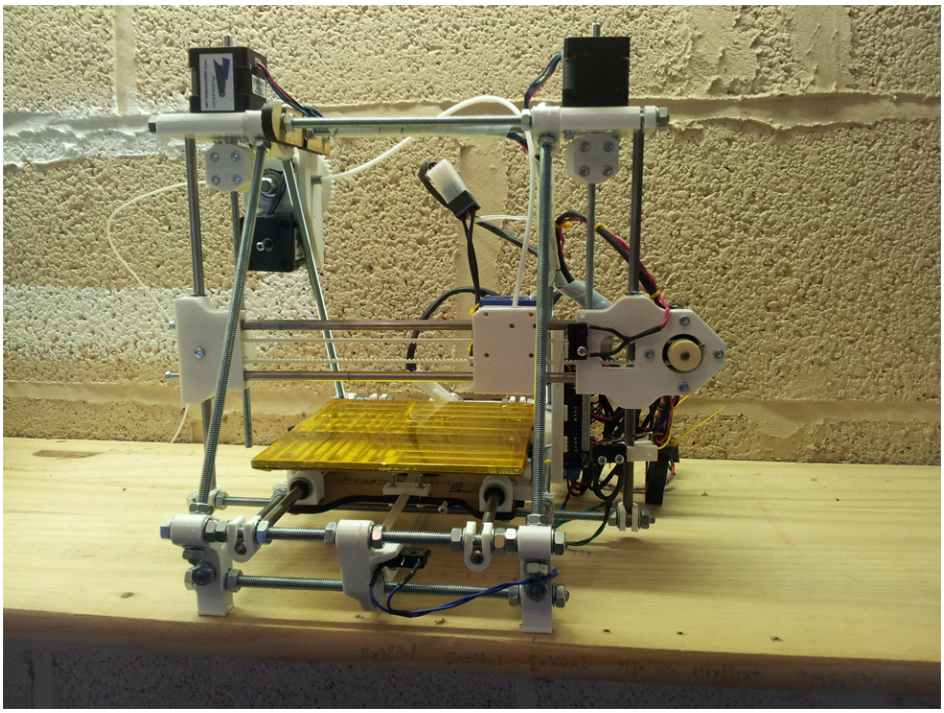
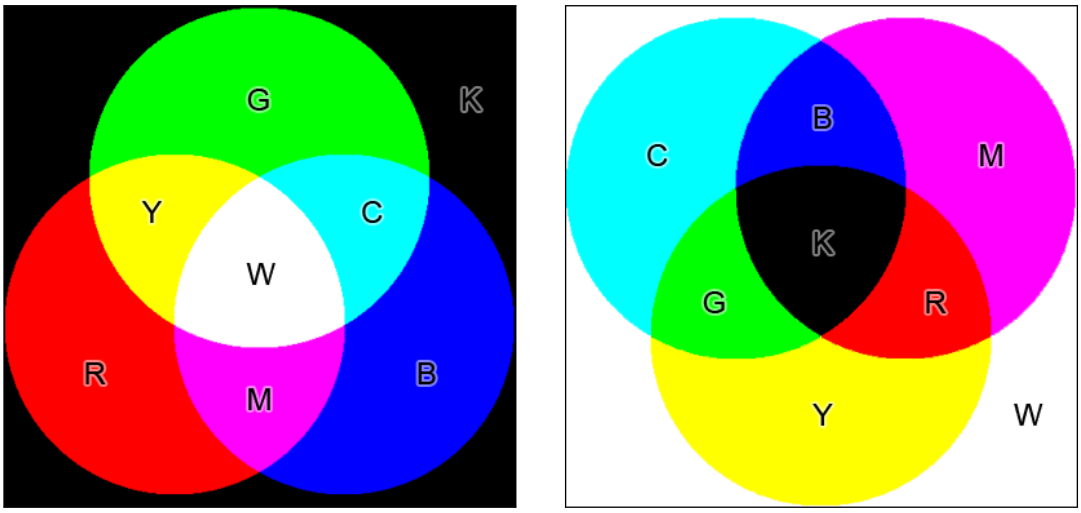


Figure 3.2.7: RepRap huxley 3D Printer using bowden cable filament drive [7]

A Huxley RepRap machine typically features a Bowden cable filament drive as shown in figure 3.2.7. As shown, the extruder drive motor is attached to the machine frame and feeds filament to the extrusion nozzle via a Bowden cable.

3.3 Colour Theory

Colour describes an impression created when light of a certain wavelength spectrum is sensed by the human eye. The visible light spectrum ranges from wavelengths of approximately 400 nanometres, which is perceived as the colour purple, to approximately 700 nanometres, which is perceived as the colour red. Combining all wavelengths of the visible light spectrum result in white light which in turn means that white light can be split up into the individual wavelengths of the visible light spectrum. As light interferes with an object, certain wavelengths are absorbed by the objects while others are reflected, refracted, scattered and diffracted. The resulting wavelengths enter the human eye where photoreceptors, which are sensitive to either short, medium or long wavelengths, detect light of certain wavelengths. The information is converted into electrochemical signals which are processed by the brain and results in the impression of the object being of certain colours [8].

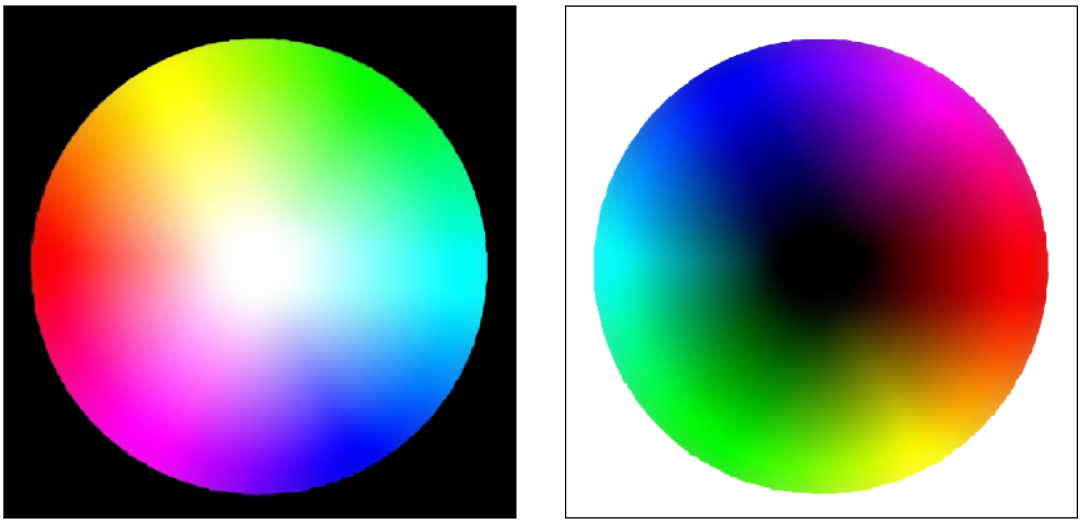


(a) Additive colour chart **(b)** Subtractive colour chart

Figure 3.3.1: Primary colour charts for additive and subtractive colour theory (**R**ed, **G**reen, **B**lue, **C**yan, **M**agenta, **Y**ellow, **K**ey (Black), **W**hite) [9]

Over the past centuries, many attempts have been made to establish a universal colour theory that helps understand the meaning, nature and interaction of colour further. Two basic models describe the additive colour theory and the subtractive colour theory. Both theories refer to a different set of primary colours which allow creating an extensive range of colours. Secondary colours result from mixing two primary colours whereas further colours result from mixing primary and secondary colours.

The additive colour theory, whose primary colours are red, green and blue (RGB), is based on the effect of adding visible light of certain wavelengths to an existing light spectrum. This principle is used to create colour on electronic displays such as computer or TV screens. The primary colours of the additive colour theory can also be referred to as the primary colours of visible light since mixing equal amounts of these colours results in white light whereas the absence of any coloured light gives black or, in terms of light, darkness. A common colour model based on the additive colour theory is the RGB colour model which is named according to its primary colours.



(a) Additive colour wheel **(b)** Subtractive colour wheel

Figure 3.3.2: Colour wheel for additive and subtractive colour theory [9]

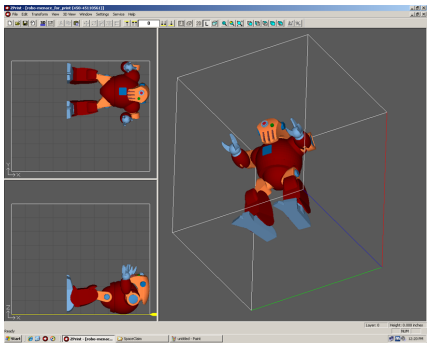
The subtractive colour theory, whose primary colours are cyan, magenta and yellow (CMY), describes the effect of mixing pigments such as is the case for mixing paint, ink and dyes. This colour theory is based on the effect of removing certain wavelengths from the existing light spectrum as it interferes with pigments. A blank colour spectrum and thus full visible light spectrum gives the colour white if it is assumed that potential pigments are applied to a white surface. Adding equal amounts of the primary subtractive colours to the colour spectrum gives black as this pigment mixture absorbs the entire wavelength spectrum of visible light. To achieve higher detail in practical applications, the colour key (K) is added to the primary colours of the subtractive colour theory which is usually achieved by using black pigments. A common colour model which is based on the subtractive colour theory is hence

the CMYK colour model.

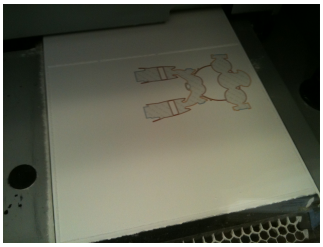
Simplified colour mixing examples are displayed in figure 3.3.1a and figure 3.3.1b for the additive colour theory and the subtractive colour theory respectively. It can be seen that the primary colours of the additive colour theory are equivalent to the secondary colours of the subtractive colour theory and vice versa. Colour wheels visualising colours resulting from additive and subtractive colour models are displayed in figures 3.3.2a and 3.3.2b respectively. Starting with the respective primary colours, these colour wheels are created by mixing increasing amounts of primary and secondary colours which gives the entire colour range that can be achieved with either model.

3.4 Colour in Additive Manufacturing

Most of the currently existing additive manufacturing machines are only capable of producing parts of a single colour. The achievable colour is contingent on the colour of the material that is used to create the parts and on the type of mechanism that is used to join the used material.



(a) CAD model of the coloured part



(b) Fabrication of coloured part

(c) Finished part with excess powder

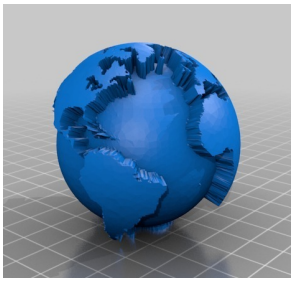
(d) Finished part after excess powder is removed

Figure 3.4.1: 3D printing process on a ZPrinter [10]

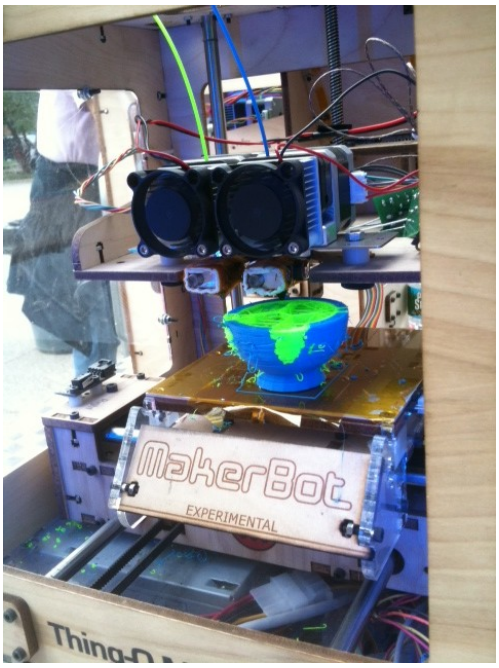
To date, ZPrinter, which is a commercial additive manufacturing machine owned by 3D Systems, is the only additive manufacturing machine that offers the possibility to create multi-coloured parts. ZPrinter is based on the process of Three-Dimensional Printing which allows depositing colour particles along with the liquid adhesive. The part production process starting with the CAD model of the coloured part and ending with the final product is shown in figure 3.4.1.

Additive manufacturing processes in which particles can be deposited selectively generally offer the potential to create objects of multiple colours. Existing AM machines based on fused filament fabrication can be equipped with multiple nozzles which allow depositing filament of different colours.

(a) STL model of blue elements



(b) STL model of green elements



(c) Fabrication of coloured part using green and blue filament

Figure 3.4.2: 3D printing process on a Makerbot [11]

Makerbot's Replicator is based on fused filament fabrication and offers a dual extrusion system which makes it possible to create two-coloured parts. The part has to be split up into two different STL files where each file represents the part of the model that are to be printed in one colour. Each STL file is then printed by either extruder so that the extruders are used

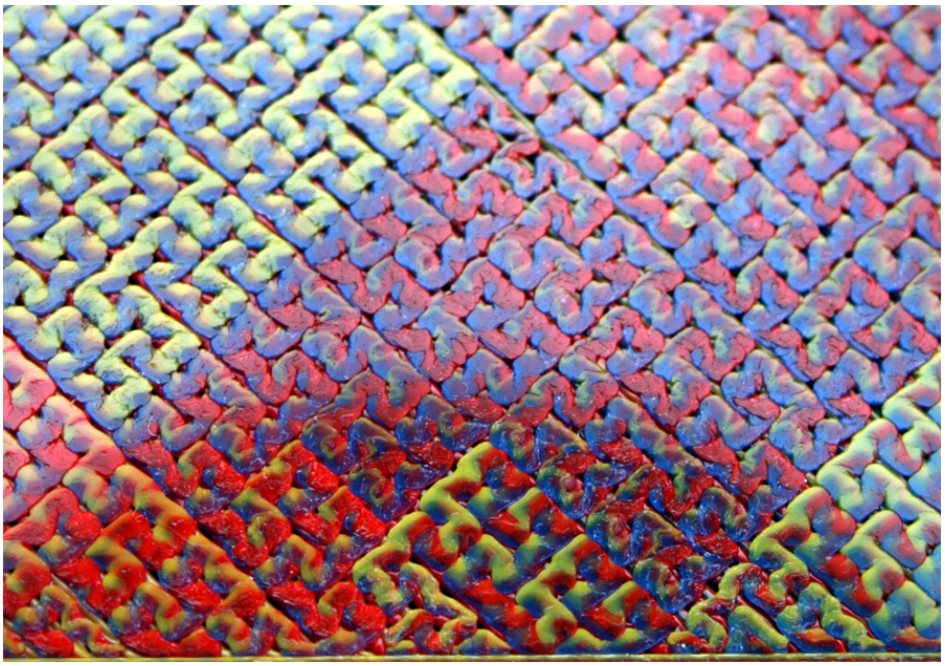


Figure 3.4.3: Part manufactured on a RepRap machine using static mixing [12]

alternately to create the dual-coloured parts. Two STL-files representing this principle and the resulting fabrication process are displayed in figure 3.4.2.

Attempts have been made to create single extrusion systems for RepRap machines which would make it possible to mix multiple filament strands in one nozzle. It has been established that static mixing systems do not create sufficient results since the extruded material clearly shows that the individual filament strands do not mix entirely but simply fuse and create a multi-coloured material output [12] [13]. A representative close-up image of a print created using static mixing is displayed in figure 3.4.3.

4 Extruder Development

4.1 Extruder Design

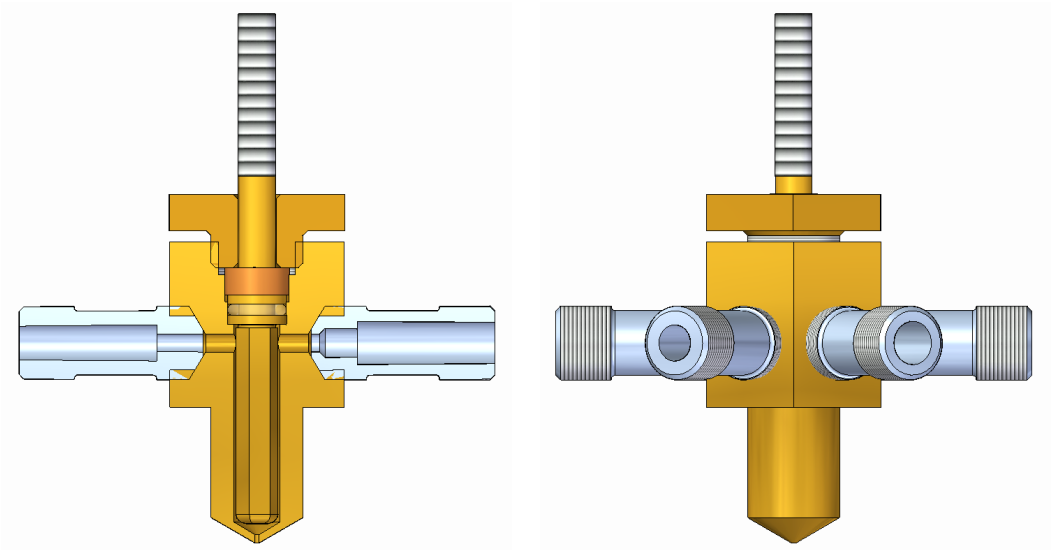
4.1.1 Required Design Features

In reference to the previously discussed literature review, several design features were identified that had to be considered during the design process. Additive manufacturing processes are based in the CMYK model but require additional white filament which results in the minimum amount of filament inputs of five. Using the colours white, cyan, magenta, yellow and black to create objects should make it possible to create parts of a wide colour range. Furthermore, a potentially additional filament input should be considered to allow the use of material of different mechanical properties. Initially, only two of the six filament inputs will be needed to evaluate the resulting design. Coloured filament that can be used with RepRap machines exist in diameters of 1.75 millimetres or 3.0 millimetres whereas softer filament can only be supplied with a 3.0 millimetres diameter. Since the extruder design shall be capable of processing coloured as well as softer filament, a universal design shall be developed that allows using filament inputs of either diameter. Mixing the input material in a static manner has been proven to give insufficient results [13] so that an active mixing process shall be used to create uniformly-coloured output material. To focus the design process on extruder features that are crucial to the filament mixing and depositing, existing extruder design features shall be implemented in the design process.
To summarise, the following design requirements have to be met:

- Six filament inputs
- Universal design that can be used with filament of either 1.75 mm or 3.0 mm diameter
- Use active mixing
- Adapt already existing and proven extruder features if possible to minimise potential causes of error
- Use already existing extruder drive mechanism if possible to focus design process on the parts more crucial to the mixing process
- As with the entire RepRap project, design extruder set up which is affordable and relatively easy to copy

4.1.2 Extrusion Nozzle Design

The previously mentioned required extrusion nozzle design features led to the design of the extrusion nozzle displayed in figure 4.1.1. This design features the main extrusion nozzle body, one mixing rod, one dynamic seal, two seal compression plates, one bush, one seal bolt and six inserts. The main extrusion nozzle body, the mixing rod, the seal compression plates and the seal bolt are machined from brass whereas the inserts are machines from stainless steel. The design process that led to this design will be explained in the following paragraphs.



(a) Cross section view of the extrusion nozzle design

(b) Front view of the extrusion nozzle design

Figure 4.1.1: Cross section view and front view of the designed extrusion nozzle

The first design requirement covered deals with the use of up to six filament inputs. To fulfil this requirement, as well as enabling the possibility to achieve a controlled filament mixing process and avoid the need to make additional changes to the used G-code to account for an asymmetric design, the filament inputs have to be located equidistant to the extrusion nozzle output. Furthermore, the filament inputs have to be spaced equidistantly to one another to reduce the possibility of the individual filament inputs interfering with one another. These filament input arrangement requirements regarding led to the intention to locate six filament inputs equidistantly on a circumference and thus the outer geometry of the extrusion nozzle. To simplify the machining process of the main body of the extrusion nozzle, this part was machined from a brass bar with a hexagonal cross section. This shape

allows locating the filament inputs on each side of the hexagonal bar.

To create a universal extrusion nozzle design which can be used with filament inputs of different diameters without imposing the necessity to perform major modifications, the individual filament inputs have to be designed according to this requirement. This led to the intention of using exchangeable attachments which are specific to different filament diameters but whose outer dimensions are identical so that their use with the main extrusion nozzle body is not restricted. A design feature used as part of an extrusion system mainly used on RepRap machines of the type *Huxley* was adapted to the specific requirements of the mixing nozzle design. This relevant design feature is located between the liquefier and the cooling mechanism on the extrusion head of a RepRap Huxley and is used to guide filament into the liquefier. This component is machined from stainless steel due to its low thermal conductivity. Similar components are designed for the use with the mixing nozzle design where the outer dimensions of this stainless steel insert are determined such that it can be used with filament of 3.0 mm diameter. The internal dimensions are such that one stainless steel insert type can be used with filament of 1.75 mm diameter and a second stainless steel insert type can be used with filament of 3.00 mm diameter.

Active mixing can be achieved by placing a moving component inside the mixing chamber of the extrusion nozzle to create artificial turbulences and sufficient shear to achieve mixing. A relatively simple solution uses a rotating mixing element driven by a motor off the RepRap electronics which allows actuating the motor when required. Once actuated, the motor has to run continuously to obtain uniform mixing which can be achieved by using a DC motor. Since this motor has to run off the RepRap electronics which are powered by a 12 Volt power supply, a 12 Volt DC motor of relatively small dimensions is required. Initial tests to create a mixing nozzle using an active mixing element driven by a 12 Volt geared DC motor have shown that a hexagonal mixing bar creates sufficient shear within the molten plastic to achieve a certain grade of mixing while the resistance inside the mixing chamber can be overcome by the DC motor. To keep the overall thermal mass of the extrusion nozzle as low as possible, the size of the mixing element and thus the size of the main nozzle body shall be kept as small as possible. The dimensions of the mixing element are determined such that a small mixing chamber volume can be achieved.

The dynamic seal in form of an O-ring capable of withstanding dynamic forces at temperatures used during the extrusion process is used to seal the top of the mixing chamber. The two seal compression plates are used to apply pressure to the O-ring so that it seals the chamber in its compressed state. The bronze bush is a self-lubricating bearing that is used to aid the rotation of the mixing element. The mixing chamber is closed with a brass seal bolt which is used to apply the pressure required to compress the O-ring and seal the mixing chamber. PTFE tape is applied to the threads of the stainless steel inserts and the brass seal bolt to seal these. Figure 4.1.2 shows an explosion view of the designed extrusion nozzle to visualise the individual components. Detailed part drawings for each of the parts can be found in the appendix.

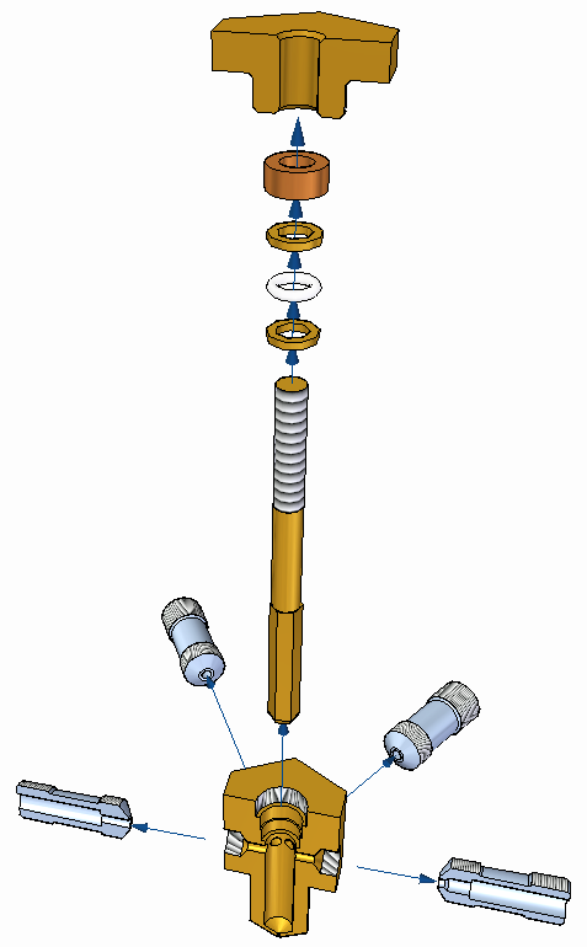


Figure 4.1.2: Exploded cross section view of the designed extrusion nozzle

The extrusion head is used in combination with a press-fit heating element similar to heating

elements used with other RepRap extrusion systems. This heating element slots over the cylinder-shaped lower part of the extrusion nozzle and was chosen since this is a non-invasive liquefying method which does not interfere with the mixing process. Furthermore, a cooling block will be clamped onto the steel inserts to ensure that heat cannot conduct along the inserts which can result in the filament softening too early thus blocking the inserts. The cooling block uses water cooling with water running through three cooling channels in the cooling block.

4.1.3 Filament Drive Mechansim

A maximum of six separate filament inputs requires six individual drive mechanisms which include motors. Positioning as many as six motors around the extruder carriage would result in a very high extruder volume and mass which might limit the possible print volume since such a setup is very likely to collide with the machine frame. To avoid this, the filament drive mechanisms were attached to the frame of the machine while being connected to the respective filament nozzle inputs via Bowden cables. An already existing extruder drive design,

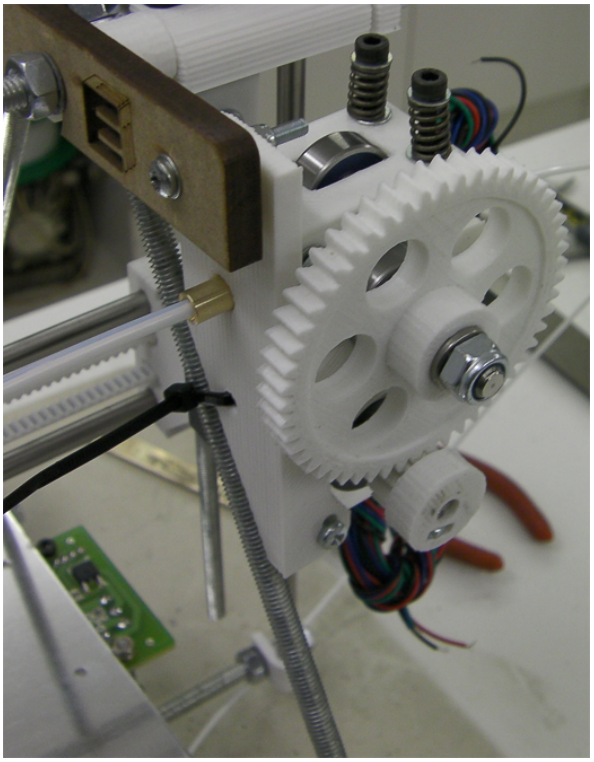


Figure 4.1.3: Drive mechanism of the RepRap Huxley bowden drive

which is displayed in figure 4.1.3, will be used for each filament input. The Bowden tube will be fixed to brass sleeves on both side which will make it possible to attach one side to the extruder drive and the other side to the extrusion nozzle.

4.2 RepRap G-code

Current G-code commands that trigger an extruder motor are of the form

```
G1 X90.6 Y13.8 E22.4 F3000
```

where *G1* starts a controlled move from the current position of x- and y-axes to the coordinates *X90.6* and *Y13.8*. The extruder extrudes a filament length of 22.4 millimetres during this move and the entire movement is performed at a feedrate of 3000 $^{\text{mm}}/_{\text{minute}}$. A G-code command of the form

```
T1
```

allows to switch from the presently used extruder to a different extruder which is extruder 1 in this example. A combination of both G-code commands makes it possible to perform an extrusion using one extruder at a time. For this section, six filament inputs will be considered for the discussion since this represents the extruder set up necessary to print using white, cyan, magenta, yellow and black filament as well as filament of different mechanical properties.

4.2.1 Required G-code Features

The extruder design will require multiple extruder drive mechanisms to drive each individual filament strand. It has to be possible to actuate these drive mechanisms simultaneously to achieve the desired results in mixing different filaments. This requires the revision of existing G-code commands so that it is possible to actuate each extruder motor when required. The revised G-code commands have to be of a form that requires only little modification to both the existing firmware as well as the existing G-code definition easily. Programmes currently used to create G-codes from STL files for RepRap machines are not capable of considering information regarding multiple extruders so that it will be necessary to create G-code required to test new extruder designs manually. Creating G-code manually means that G-code is not created from an STL file but is written by manually describing the movements and

extrusions the RepRap machine has to perform to create a certain object. It therefore has to be possible to create new G-codes for simple test parts or to modify existing G-codes easily. Furthermore, it has to be possible to implement G-codes for as many or as little different extruders as possible.

4.2.2 Revised G-code Commands

The already existing G-code command of the form

```
G1 X90.6 Y13.8 E22.4 F3000
```

has been adapted such that additional information regarding extrusion rates for multiple extruders were included. The individual extrusion rates have to sum up to the extrusion rate required if only one extruder was used during the printing process which is 22.4 millimetres in the mentioned example. Different options were considered in this process which have different advantages and disadvantages and will be discussed in the following paragraphs.

4.2.2.1 1st G-code Solution

One potential G-code modification involves introducing six new extrusion rates which replace the existing single-extruder extrusion rate indicated by the letter *E*. Including the additional extrusion rates in the mentioned G-code example would result in G-code commands of a form similar to

```
G1 X90.6 Y13.8 A10.2 B2.1 C3.1 D5.5 H1.5 I0.0 F3000
```

where the extrusion rates indicated by the letters *A*, *B*, *C*, *D*, *H* and *I* give the respective extrusion rates for each individual extrusion drive mechanism. These rates add up to the single-extruder extrusion rate. The revised G-code instructs a RepRap machine to move to the coordinates *X90.6* and *Y13.8* while extruder *0* extrudes 10 mm of filament, extruder *1* extrudes a filament length of 2.1 mm, and so on. In this solution, the single-extruder extrusion rate is not needed anymore and calculations regarding the extrusion rates for each of the six extrusion drives are calculated when generating the G-code. To modify existing G-codes, the used single-extruder extrusion values have to be determined to create new extrusion values for up to six different extrusion drives so that the single-extruder value can be deleted from

the G-code output file.

Adapting the RepRap firmware so that it is capable of processing this G-code solution is likely to be a rather complex process since information regarding extrusion rates is required at many stages in the firmware. This includes checking whether reasonable amounts of filament are about to be extruded or evaluating movements that are about to be performed. Implementing information regarding the modified G-code commands in the firmware would therefore be a lengthy process.

4.2.2.2 2nd G-code Solution

A different idea to achieve the modified G-code is based on the intention to retain the already existing G-code and define additional commands that hold information regarding the different extruder drive mechanisms. The additional G-code commands would contain percentage values indicating the percentage by which each extrusion drive contributes to the single-extruder extrusion rate. A respective G-code command could be of the form

G1 X90.6 Y13.8 E22.4 A0.455 B0.094 C0.138 D0.246 H0.067 I0.0 F3000

where the extrusion components, which again are indicated by the letters A, B, C, D, H and I, add up to an overall value of *1*. This solution still requires the use of the extrusion value E since calculations regarding the actual extrusion rates achieved by each extruder drive mechanism is calculated by the RepRap firmware. Adapting existing G-codes to the revised G-code definition can be done easily since the single-extrusion rate is not relevant when determining the extrusion ratios for each of the potentially six extruders. Combining the overall extrusion rate with the individual ratios for each extruder will be performed by the RepRap firmware at a later stage. Furthermore, the single-extruder extrusion rate is not of importance when already existing G-code files are adapted to the new G-code commands and can simply be ignored during this process.

The RepRap firmware can be easily modified to being able to process this G-code solution since it is only necessary to implement required modifications in parts of the firmware that are responsible for the actual filament extrusion process. Overall, this G-code solution seems to require only minor adaptions to existing programmes so that this G-code command defini-

tion will be used to print multi-coloured test samples using the newly developed nozzle design.

4.2.3 G-code Processing

Creating a G-code from an STL file with any programme available will not offer the possibility to insert extrusion ratios A and B which is necessary to test the mixing nozzle design. To solve this problem a C++ programme was written which uses an existing G-code to read the coordinates X and Y of the part that is to be printed, process this information along with a user input regarding the desired colour mixing format and output the resulting modified G-code including respective extrusion ratios. This programme, whose code is listed in the appendix, allows creating parts using one single colour, a specific constant mixing ratio and linear or sinusoidal colour transition throughout the object.

4.3 RepRap Firmware

The RepRap firmware is used to process G-code received from a G-code file or a computer RepRap user interface such as *Printrun*. The firmware analyses the G-code and initiates the process the RepRap machine is required to perform according to the G-code. This process can involve actuating any of the axes motors, extruding filament and setting, reading or controlling temperatures of the extruder hot end and the heated bed.

4.3.1 Required Firmware Features

The firmware section responsible for switching between selected extruders and initiating actions performed by the selected extruder is listed in table 4.3.1. Cases for only one or more than two extruders being available to the machine are not listed since these are not important for testing the mixing process of two filament inputs. However, the displayed firmware code can easily be adapted to up to six or more filament inputs so that potential further developments are not impeded by the firmware currently in use. The firmware extract shows that actions are initiated depending on which extruder is the currently used active extruder while the inactive extruders are neglected. This section therefore has to be adapted such that it is possible to drive two extruders simultaneously.

stepper.h

```
#if EXTRUDERS >2
  [...]
#elif EXTRUDERS >1
  #define WRITE_E_STEP(v) { if(current_block->active_extruder == 1)
    { WRITE(E1_STEP_PIN, v); } else { WRITE(E0_STEP_PIN, v); }}
  #define NORM_E_DIR() { if(current_block->active_extruder == 1)
    { WRITE(E1_DIR_PIN, !INVERT_E1_DIR); }
    else { WRITE(E0_DIR_PIN, !INVERT_E0_DIR);}}
  #define REV_E_DIR() { if(current_block->active_extruder == 1)
    { WRITE(E1_DIR_PIN, INVERT_E1_DIR);}
    else { WRITE(E0_DIR_PIN, INVERT_E0_DIR); }}
#else
  [...]
#endif
```

Table 4.3.1: RepRap firmware extract from file stepper.h

The firmware section listed in table 4.3.2 is responsible for actuating the extrusion process for a certain amount of stepper motor steps required performing the desired extrusion process. As seen with the previous firmware section listed in table 4.3.1, the expression *WRITE_E_STEP(v)* is defined such that an extrusion process is initiated for the selected extruder. Calling this definition to perform an extrusion as listed in table 4.3.2 can therefore be done without specifying which extruder will be used to perform for this process. This part of the firmware has to be modified since the individual extruder motors used as part of the mixing set up will most likely have to be actuated for different amounts of steps or extruded filament.

Further required firmware modifications cover retrieving extrusion ratios for the individual extrusion drives from the G-code and calculating the resulting amount of steps that have to be performed by each extruder stepper motor to achieve the desired extrusion.

stepper.cpp

counter_e += current_block->steps_e;
if (counter_e >0) {
WRITE_E_STEP(HIGH);
counter_e -= current_block->step_event_count;
WRITE_E_STEP(LOW);
count_position[E_AXIS]+=count_direction[E_AXIS]; }

Table 4.3.2: RepRap firmware extract from file stepper.cpp

4.3.2 Revised Firmware

The first firmware modifications performed allow actuating two extruder stepper motors simultaneously which was achieved by adapting the firmware section listed in table 4.3.1 which led to the revised firmware code which is listed in table 4.3.3. It can be seen that by using two extruders, each extruder stepper motor will be actuated according to the amount of steps each motor has to perform.

stepper.h
#if EXTRUDERS >2 [...] #elif EXTRUDERS >1 #define WRITE_E0_STEP(v) { WRITE(E0_STEP_PIN, v); } #define WRITE_E1_STEP(v) { WRITE(E1_STEP_PIN, v); } #define NORM_E_DIR() { WRITE(E1_DIR_PIN, !INVERT_E1_DIR); WRITE(E0_DIR_PIN, !INVERT_E0_DIR); } #define REV_E_DIR() { WRITE(E1_DIR_PIN, INVERT_E1_DIR); WRITE(E0_DIR_PIN, INVERT_E0_DIR); } #else [...] #endif

Table 4.3.3: Modified RepRap firmware file stepper.h

Further modifications were made to the firmware section responsible for retrieving information regarding x-, y- and z-coordinates, feedrate and extrusion rate from the G-code which is listed in table 4.3.4. This section was modified such that the firmware furthermore checks for extrusion ratios labelled with the letters *A* and *B*. Information regarding x-, y- and z-coordinates *X, Y* and *Z*, feedrate *F*, extrusion rate *E* and extrusion ratios *A* and *B* is then used to initiate axes movements and extrusion by calling the function *prepare_move()*.

Marlin_multi.pde

```
FORCE_INLINE void get_coordinates() {
  for(int8_t i=0; i <NUM_AXIS; i++) {
    if(code_seen(axis_codes[i])) destination[i] = (float)code_value() +
      (axis_relative_modes[i] ||relative_mode)*current_position[i];
    else destination[i] = current_position[i]; }
  if(code_seen('F')) {
    next_feedrate = code_value();
    if(next_feedrate >0.0) feedrate = next_feedrate; }
  if(code_seen('A')) extrusion_rate[0] = code_value();
  if(code_seen('B')) extrusion_rate[1] = code_value(); }

void prepare_move() {
  [...]
  plan_buffer_line(destination[X_AXIS], destination[Y_AXIS], destination[Z_AXIS],
    destination[E_AXIS], extrusion_rate[0], extrusion_rate[1],
    feedrate*feedmultiply/60/100.0, active_extruder);
  [...] }
```

Table 4.3.4: Modified RepRap firmware file Marlin_multi.pde

This function was modified such that it processes information regarding extrusion ratios *A* and *B* further as shown in table 4.3.5. This information is furthermore made accessible to a different part of the firmware that is responsible for initiating the actuation of the extruder motors which is listed in table 4.3.6 and shows the changes made to the original firmware code listed in table 4.3.2. The steps each extruder stepper motor has to be actuated for are

planner.cpp

void plan_buffer_line(const float &x, const float &y, const float &z, const float &e,
const float &a, const float &b, float feed_rate, const uint8_t &extruder) {
[. . .]
block->e0_proportion = a;
block->e1_proportion = b;
[. . .] }

Table 4.3.5: Modified RepRap firmware file planner.cpp

calculated and each of the extruder stepper motors is actuated such that the amount of filament required to achieve the desired mixing result is extruded. Having made the mentioned changes to the RepRap firmware now allows to use two extruders attached to a RepRap machine simultaneously. Furthermore, these modifications can be adapted easily to enable to use of six separate extrusion drives so that it is possible to manufacture parts on a RepRap machine using six individual filament inputs to achieve complex colour mixing results.

stepper.cpp
counter_e0 += current_block->steps_e*current_block->e0_proportion;
counter_e1 += current_block->steps_e*current_block->e1_proportion;
if (counter_e0 >0) {
WRITE_E0_STEP(HIGH);
counter_e0 -= current_block->step_event_count;
WRITE_E0_STEP(LOW);
count_position[E_AXIS]+=count_direction[E_AXIS]; }
if (counter_e1 >0) {
WRITE_E1_STEP(HIGH);
counter_e1 -= current_block->step_event_count;
WRITE_E1_STEP(LOW);
count_position[E_AXIS]+=count_direction[E_AXIS]; }

Table 4.3.6: Modified RepRap firmware file stepper.cpp

5 Design Evaluation

5.1 Extruder Assembly

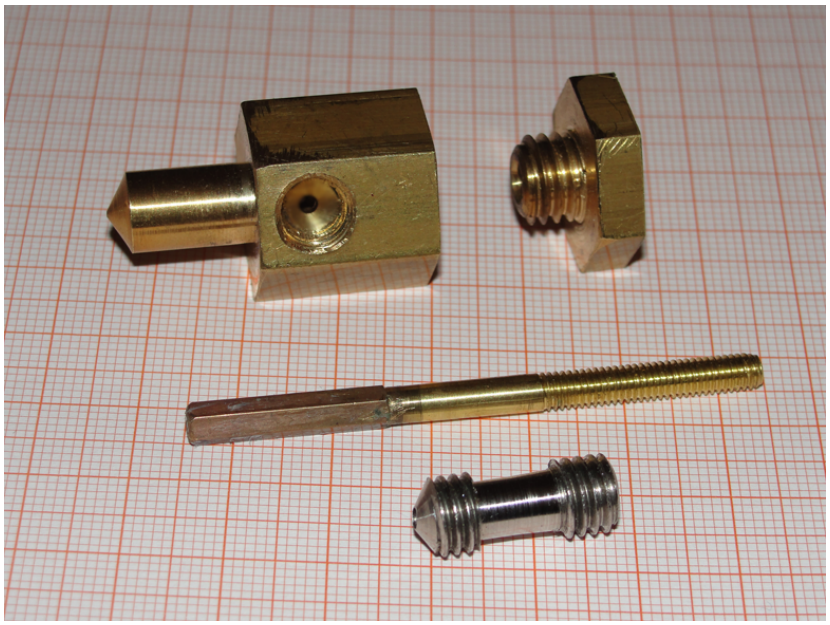


Figure 5.1.1: Machined mixing nozzle parts

The mixing nozzle extruder design consists of several parts that have to be machined includes components made from brass or stainless steel. The respective parts used for the mixing nozzle are displayed in figure 5.1.1 which shows the brass nozzle body, the brass seal bolt, the brass mixing rod and one of the stainless steel inserts.

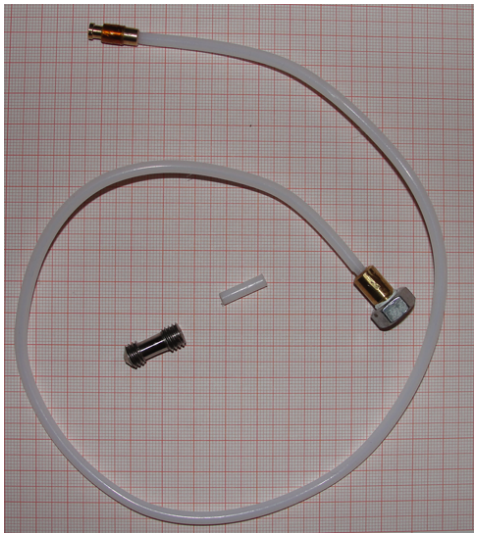


Figure 5.1.2: Bowden drive components

Along with an O-ring, two brass seal compression plates and one bronze bush, these components can be assembled to the mixing element of the design. PTFE tubing is used to form the Bowden cable to which brass fittings have been attached, as shown in figure 5.1.2. One of the brass fitting fits into the extruder drive mechanism whereas the other brass fitting is attached to the steel insert in the mixing nozzle via a steel nut. This nut only holds the function of joining the stainless steel insert and the brass fitting. A short bit of PTFE tubing is fitted into the end of the stainless steel insert facing away from the mixing nozzle which will be used to help guide the filament. The resulting mixing nozzle assembly is displayed in figure 5.1.3. As can be seen in this figure, the previously mentioned water cooling block

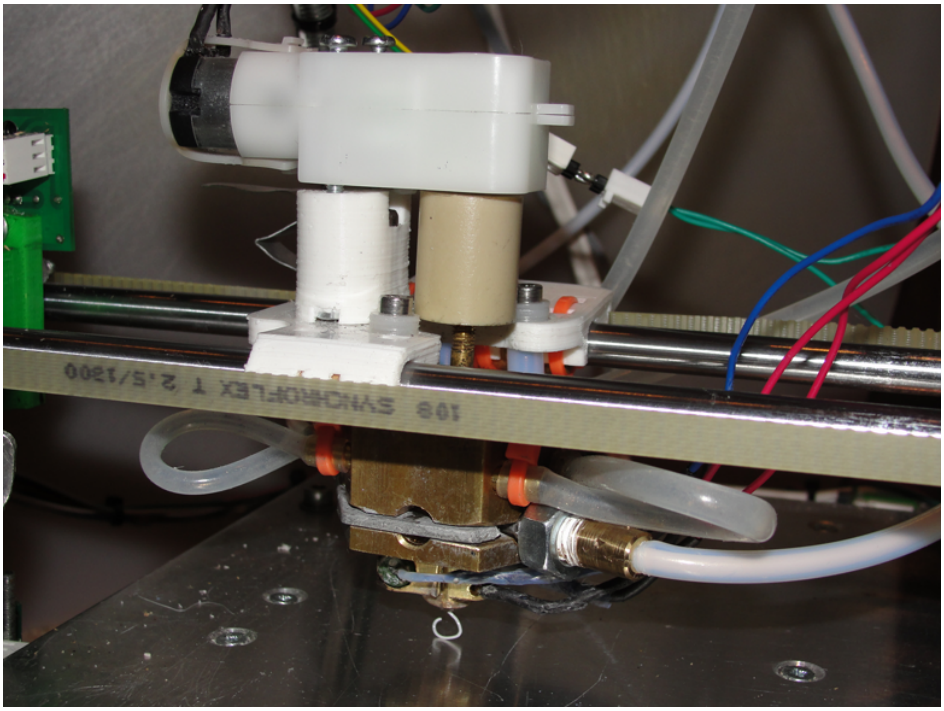


Figure 5.1.3: Mixing nozzle assembly

has been attached to the mixing nozzle while the mixing rod is connected to the DC motor via a peek coupling. The DC motor is also supported by a component that is attached to the extruder carriage to minimise any potential load the motor might be subjected to which is not caused by the mixing process itself. The mixing nozzle is fitted with two filament inputs which will be used to print samples using filament of two different colours and later on filament of different mechanical properties. The material that will be used to create these samples is PLA since filament of this material is available in many different colours but also different mechanical properties so that it is ensured that the material of different mechanical

properties will mix.

5.2 Multi-coloured Extrusion

Several parts were manufactured using the mixing nozzle set up with two filament inputs. To evaluate the quality and repeatability of the mixing process, initial tests were performed using filament of different colours, i.e. blue and white. Square test pieces were printed using a wide range of mixing ratios ranging from a mixture consisting of 0 % white and 100 % blue to a mixture consisting of 100 % white and 0 % blue. The mixing ratio was altered for each print using steps of 5 %. The respective G-code was processed using the open-source software *Slic3r* in combination with the programme written to modify an existing G-code according to respective colour mixing requirements. The resulting parts are displayed in figure 5.2.1. The upper two parts have been printed using a single-filament extruder and will be used as reference. The remaining parts have been printed using a specific fixed extrusion ratio as indicated by the labelling on the figure. Starting with the part of the ratio 100 % white and 0 % blue, it can be seen that the individual parts are printed as expected with the amount of blue increasing along with the theoretical change in extrusion ratio. Almost all the parts created using the mixing nozzle feature a stripe near their right side whose colour slightly differs from the main part colour. Since this feature is present in the majority of the parts which makes this feature repeatable, it can be assumed that this is caused by the combination of the G-code used and the mixing nozzle set up. Of further importance is the fact that this stripe does not occur in the same area on each layer of the objects but on every other layer. The top layer of each of the displayed parts was started from their right side so that the stripe occurs shortly after the extruder carriage has lifted by one layer height to then resume extruding filament to print the next layer. This fact indicates that the occurrence of this stripe correlates with the start of a new layer which in turn means, that no filament is extruded for the short amount of time it takes the machine to move to a new layer. During the process of filament being actively fed into the nozzle, constant amounts of filament enter the nozzle. When the process if actively feeding filament stops, already molten plastic continues to trickle from the filament input channels into the mixing chamber. When these small amounts of filament do not represent the desired mixing ratio, stripes occur in the printed parts.

An attempt was made to eliminate this problem by writing a C++ programme that can create G-code for a cube similar to the part processed with *Slic3r*. The programme code can be viewed in the appendix. The C++ programme is written such that fewer periods exist during which no filament is supposed to be extruded. This has been done by eliminating most of the movements performed along either of the machine axes where no filament extrusion occurs simultaneously. Furthermore, filament will be retracted in-between printing the individual layers when the z axis move is performed. This retraction will not be performed at the mixing ratio used during the printing process but will be performed by the same amount. Parts printed from G-code generated with the C++ programme are shown in figure 5.2.2. Comparing the parts printed using G-code generated with *Slic3r* to parts printed using G-code generated with the C++ programme indicates that a more uniform colour distribution can be achieved by using G-code generated with the C++ programme. Areas of slightly different colour can still occur but these are less prominent if present. This suggests that the settings resulting in ideal prints depend on the mixing ratio that is used but since the settings used with the C++ programme have already resulted in improved mixing results, these settings will be used for further tests.

Further test parts were manufactured to test the mixing quality by printing colour transitions which should ideally change gradually from white on the left side to dark blue on the right side. The results are displayed in figure 5.2.3. The upper four prints were printed from G-code created using the C++ programme. It can be seen, that the colour transition seems to be shifted so that white and dark blue occur slightly delayed. This information was used to adapt the C++ programme for gradually changing colour along the x-axis of the machine. The resulting programme was again used to generate G-code which in turn is used to create new parts. The resulting prints are displayed as the lower two prints in figure 5.2.3. Comparing the parts created from the modified C++ programme to the parts created from the original programme show a clear improvement with the original colours white and dark blue now occurring at the sides of the prints.

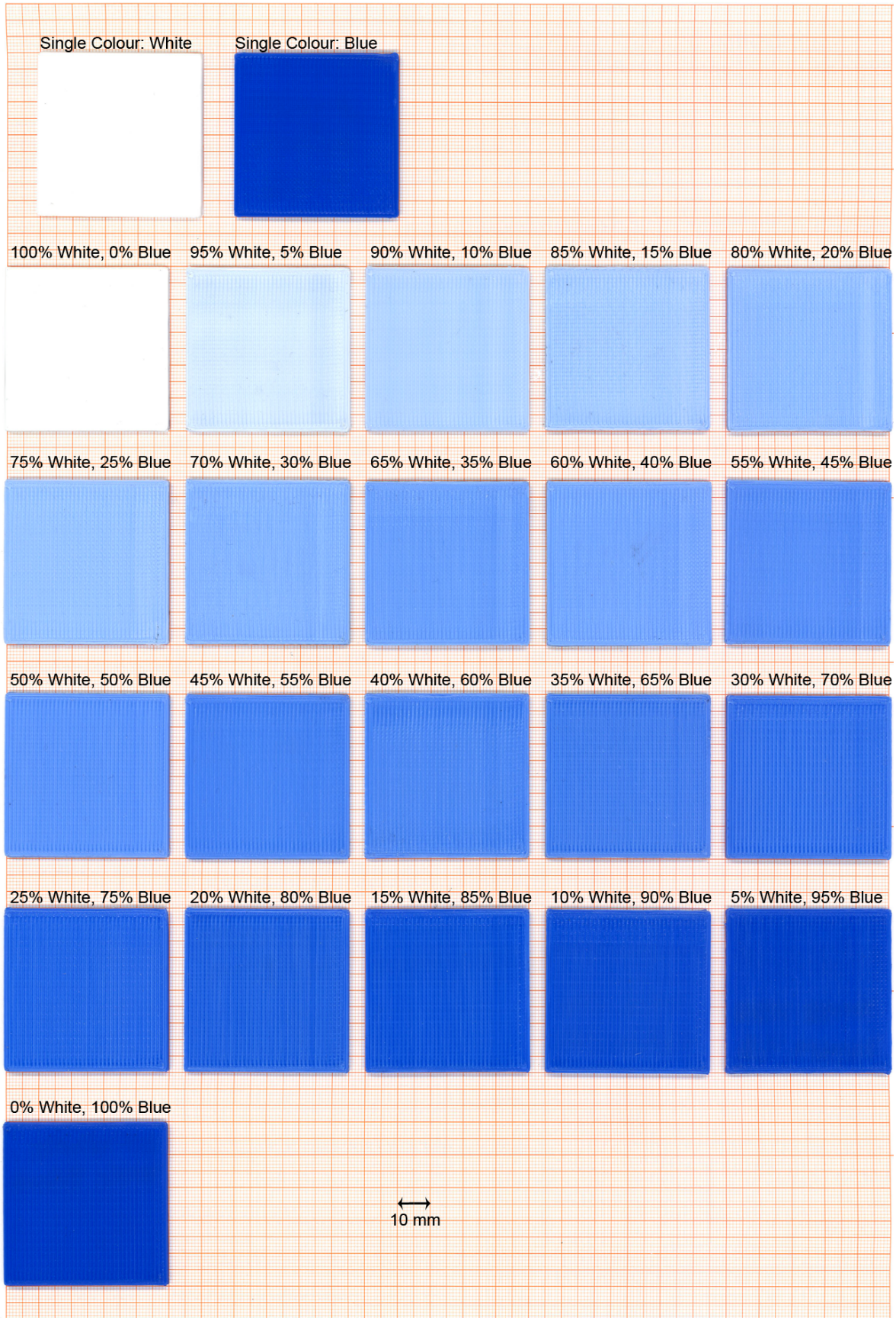


Figure 5.2.1: Sample parts printed using G-code processed with *Slic3r*

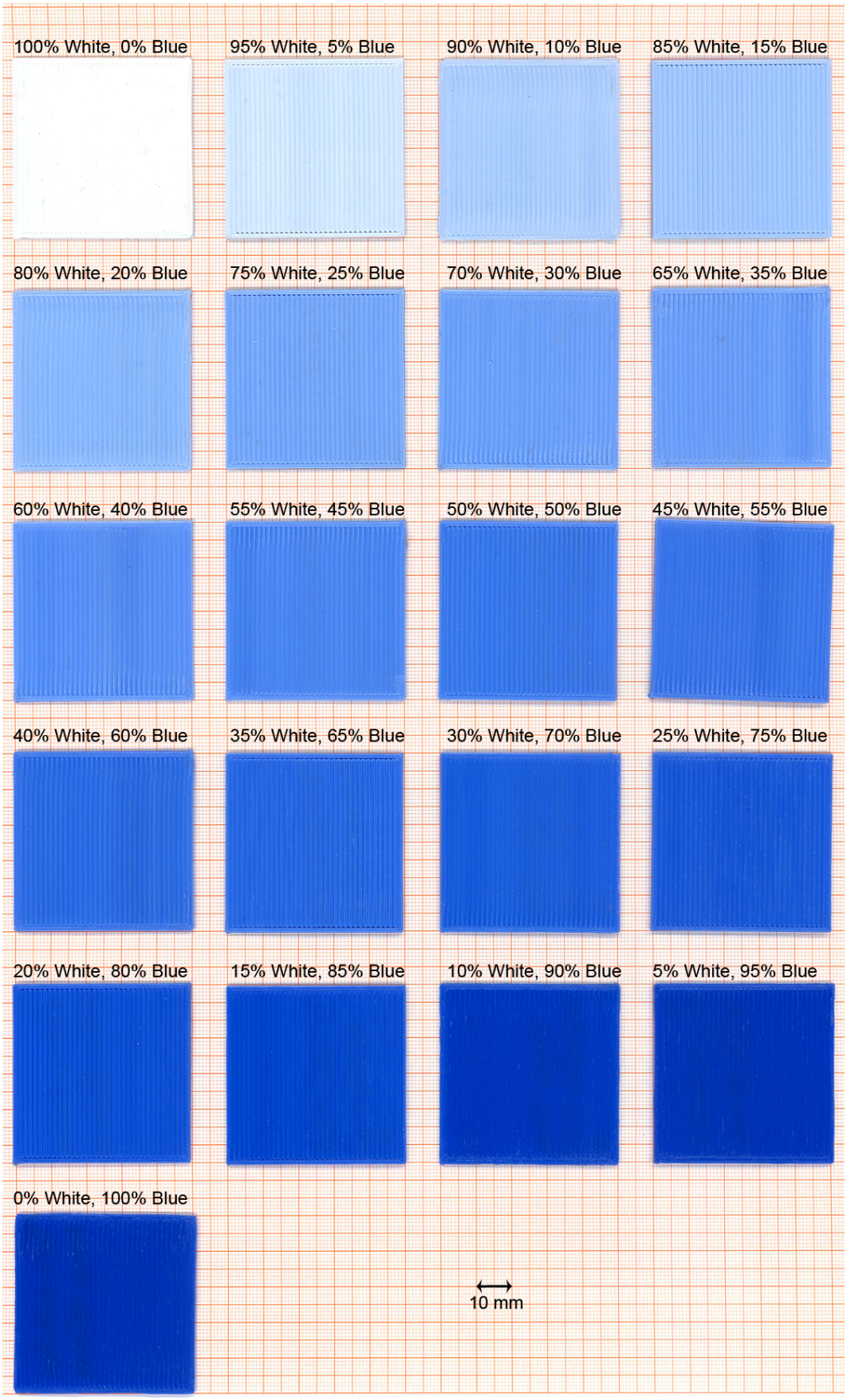


Figure 5.2.2: Sample parts printed using G-code processed with C++ programme

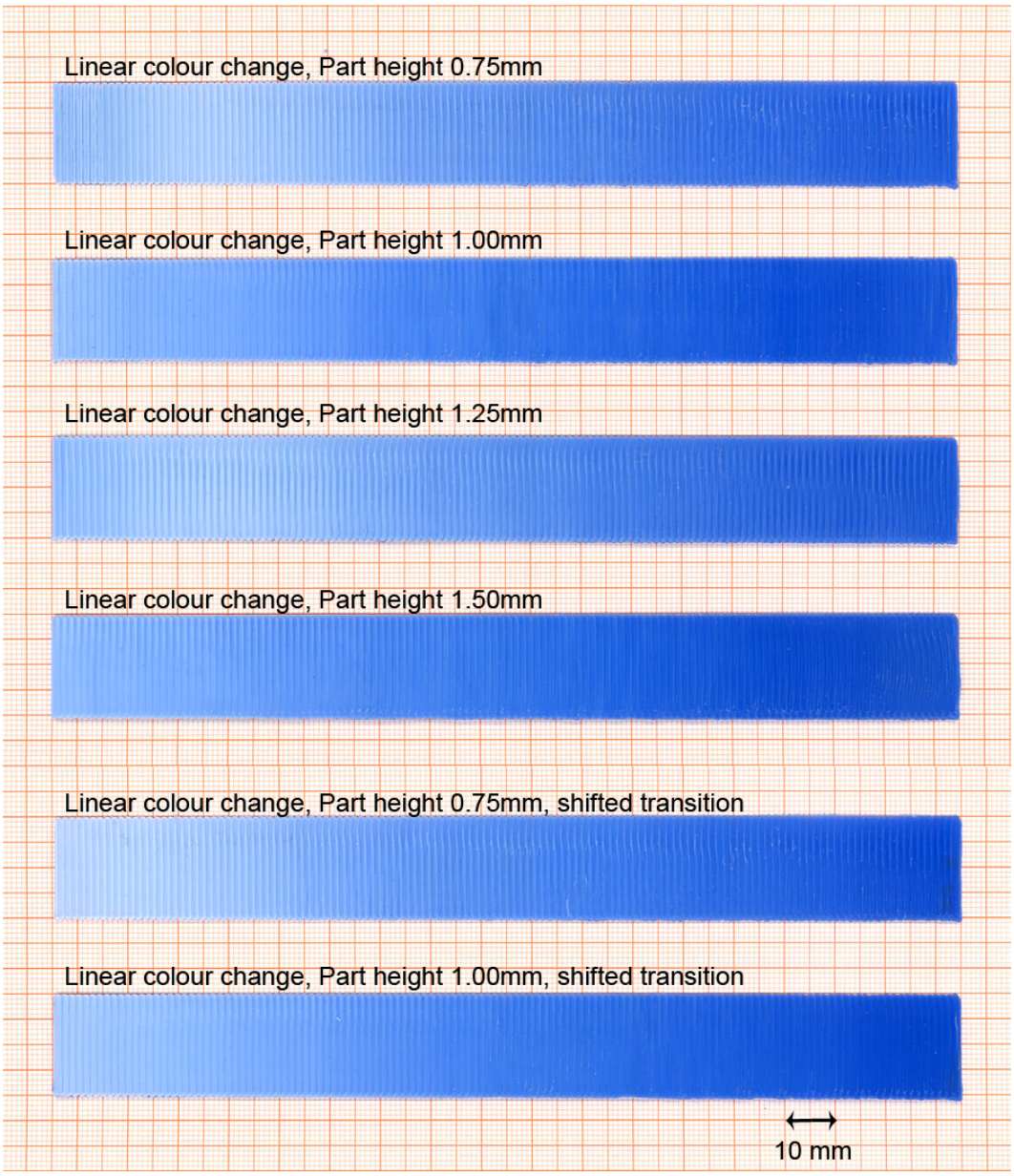
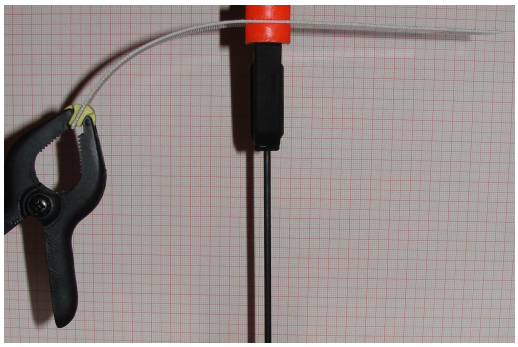


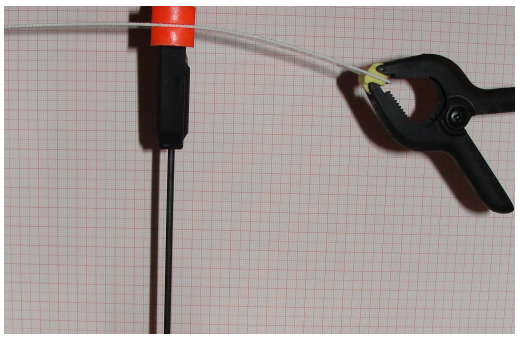
Figure 5.2.3: Colour transition parts printed using G-code processed with C++ programme

5.3 Extrusion With Varying Mechanical Properties

The C++ programme used to generate G-code has been used to print parts of varying mechanical properties. G-code, very similar to the code used to create the colour transition parts, was used to print parts that have a continuous transition of mechanical properties. One of these samples is displayed in figure 5.3.1. The part is fixed in its centre which should consist of an equal amount of hard and soft PLA and a certain weight is attached to either side of the print. Figure 5.3.1a shows the displacement of the print side mainly consisting of soft PLA while figure 5.3.1b shows the displacement of the side of the print mainly consisting of hard PLA. A clear difference in the indicated displacement is visible which show that achieving a graduate property change from hard to soft PLA has been successful. Figure 5.3.2 displays a similar set up in which a part only consisting of hard PLA is used. This can be used as a reference for the observed displacement of the printed part shown in figure 5.3.1b. It can be seen that the observed displacement for both parts is very similar. The print containing elements of soft PLA shows a slightly higher displacement than the part that only consists of hard PLA. This is due to the fact that the mixed print contains quantities of soft PLA which become more concentrated towards the centre.



(a) Displacement after clamping the print centre and attaching a weight to the soft side



(b) Displacement after clamping the print centre and attaching a weight to the soft side

Figure 5.3.1: Displacement of soft and hard side of the property transition print

Figure 5.3.3 shows the same part whose soft (figure 5.3.3a) or hard side (figure 5.3.3b) is fixed while a weight is attached to the respective other side. Both versions show a very different displacement which again proves that a gradual change in mechanical properties has been achieved. Fixing the soft side and applying a weight to the hard side results in a very high displacement whereas fixing the hard side and attaching a weight to the soft side results in a much lower displacement.

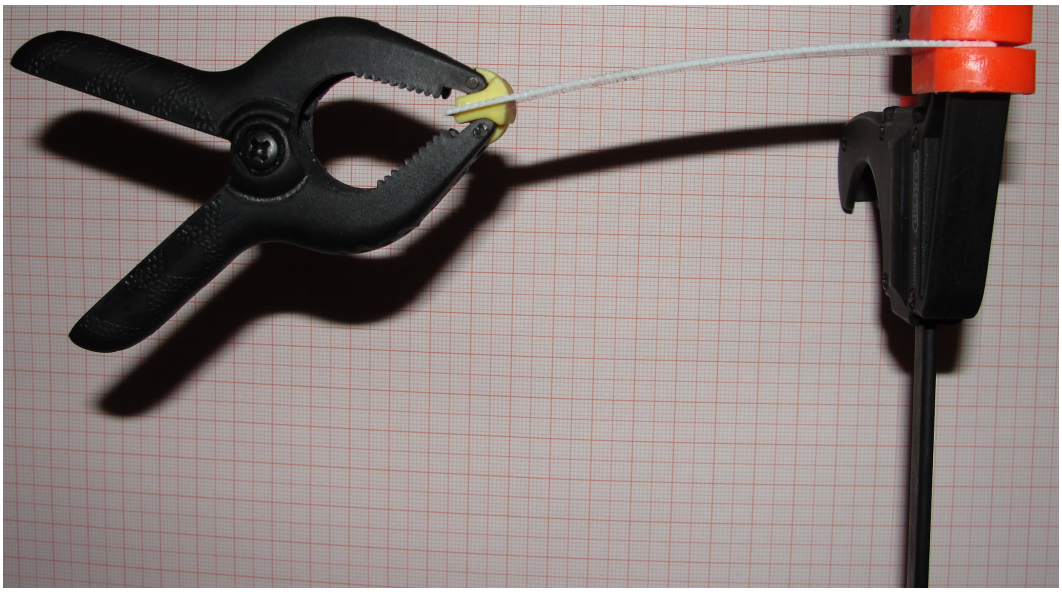
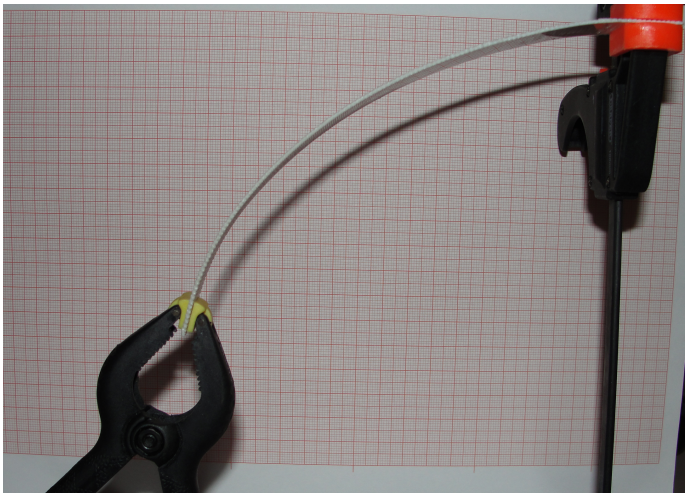


Figure 5.3.2: Displacement of reference hard print

Unfortunately, no further and more in-depth tests could be performed since a required test set up was not available.

(a) Displacement after clamping the soft side and attaching a weight to the hard side



(b) Displacement after clamping the hard side and attaching a weight to the soft side

Figure 5.3.3: Displacement of the entire property transition print

6 Conclusion

It has been shown that it is possible to design, machine and assemble a successfully working RepRap extruder that is capable of processing two individual filament strings which are actively mixed in the nozzle. Several programmes had to be adapted or re-written before they could be used in combination with the developed extrudesion system since existing programmes were not capable of dealing with multiple extruder drives which had to be actuated simultaneously. Once the software modifications were applied, it was possible to print parts of any desired fixed mixing ratio as well as objects whose colour changed gradually.

Evaluating the printed object has shown that the results massively depend on the settings used with the RepRap machine. If settings were chosen correctly, the resulting manufactured objects were of uniform colour whereas objects manufactured with less ideal settings had small areas where the resulting colour differed from the desired colour. Printing objects whose colour transitioned throughout the print showed that the designed extrusion head operates with a lag time which has to be taken into account when attempting to achieve a certain colour or material property transition in the printed part. The lag time might in turn mean that a prediction of the colour or material property distribution throughout the printed part is not possible since it is likely that the mixing process is not operated on a *First In First Out* basis.

Further tests were performed using material of different mechanical properties in the form of hard PLA filament and soft PLA filament. The resulting objects again showed that a thorough mixing could be achieved which is consistent with the theoretical distribution of mechanical properties. The mixing results were similar to those obtained when mixing coloured filament so that the use of either does not seem to impose any major problems.

7 Future Work

During the evaluation process of the designed extrusion head, only a limited amount of tests could be conducted so that one potential task for future work is to perform further extensive testing of both mixing filament of different colours and of different mechanical properties. Using the findings from these tests, the extruder design can then develop further to improve the mixing capability and reliability of the design further.

Furthermore, additional filament inputs can be used with the nozzle design to potentially achieve full-colour three-dimensional prints. An essential problem one will encounter is the fact that the STL file format which is currently used to create G-code does not support information on colour so that a different file format will have to be used to successfully perform full-coloured prints.

8 References

References

[1] Gibson, I., Rosen, D. W., Stucker, B.: *Additive Manufacturing Technologies* (2010), Springer Science + Business Media, USA

[2] CustomPartNet: *Rapid Prototyping*, http://www.custompartnet.com, retrieved 01.10.2012

[3] Ranellucci, A.: *STL-to-GCODE translator for RepRap printers*, http://slic3r.org/, retrieved 10.10.2012

[4] –: *RepRap Wiki*, http://www.reprap.org, retrieved 28.09.2012

[5] –: *Stratasys*, http://www.stratasys.com/, retrieved 30.09.2012

[6] –: *MakerBot*, http://www.makerbot.com/, retrieved 05.10.2012

[7] –: *eMaker*, http://www.emakershop.com/, retrieved 05.10.2012

[8] –: *Causes Of Color*, http://www.webexhibits.org, retrieved 15.09.2012

[9] –: *Colour Theory*, http://www.color-theory-phenomena.nl, retrieved 15.09.2012

[10] –: *3D Printer Applications for Creation*, http://3dprinters.biz, retrieved 20.09.2012

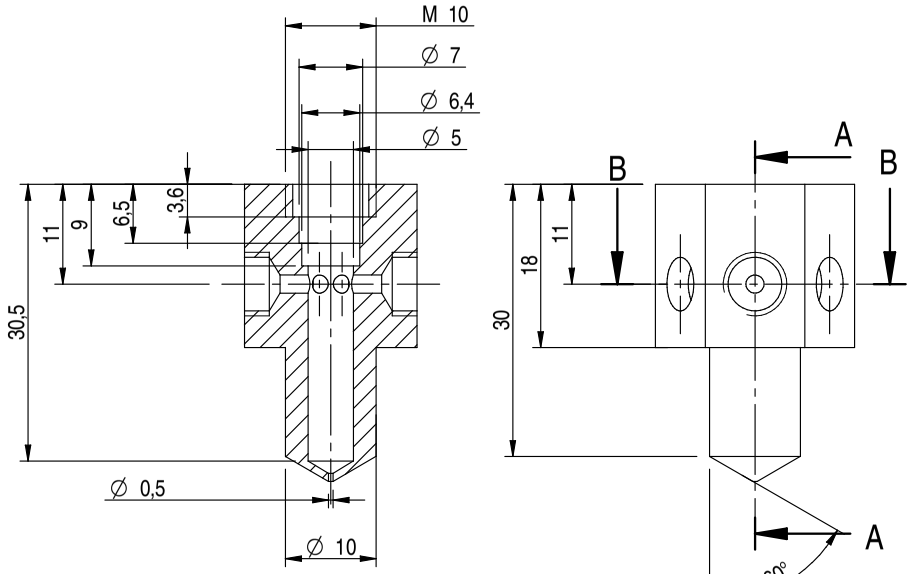
[11] –: *Dual-Head Two-Color MakerBots Are Coming!*, http://blog.makezine.com, retrieved 20.09.2012

[12] –: *3-way Quick-fit Extruder and Colour Blending Nozzle*, http://richrap.blogspot.co.uk, retrieved 20.09.2012

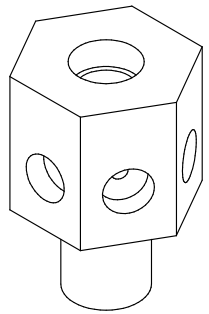
[13] Corbett, J.: *RepRap Colour Mixing Project* (2012), University of Bath, Bath

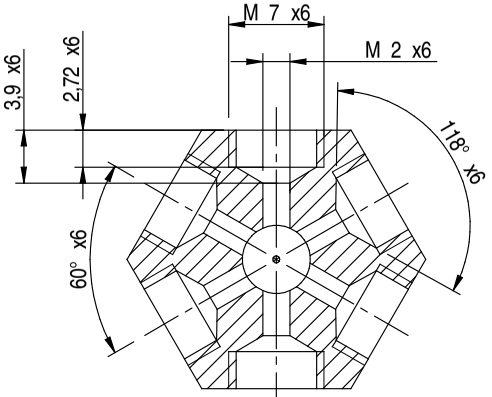
9 Appendices

9.1 Part Drawings



SECTION A-A
2:1





SECTION B-B
3:1

DO NOT TAKE MEASUREMENTS FROM DRAWING PRINTS. IF IN DOUBT: ASK.

ISSUE	DATE	DRAWN	APPROVED	DETAILS OF CHANGES

REVISION HISTORY

MATERIAL

PROTECTIVE FINISH

DIMENSIONS IN MM | SCALE AT A3 SIZE:

DRAWN IN ACCORDANCE WITH BS8888

TOLERANCES UNLESS OTHERWISE STATED
LINEAR ± 0.1mm
ANGULAR ± 2deg
SURFACE TEXTURE ✓

REMOVE ALL BURRS AND SHARP EDGES
0.5 MAX. RADIUS OR 45°CHAMFER

UNIVERSITY OF BATH
DEPARTMENT OF MECHANICAL ENGINEERING

ACTIVITY	NAME	DATE
DRAWN	PIA TAUBERT	21/10/12
CHECKED		
APPROVED		

PROJECT: Continuously-variable material properties in RepRap 3D printing

TITLE: Nozzle

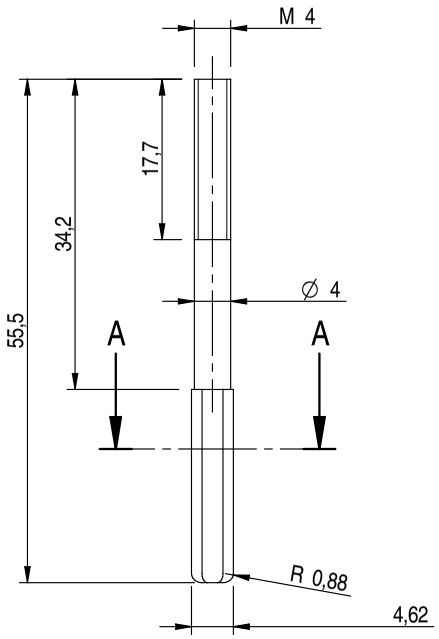
THIRD ANGLE PROJECTION

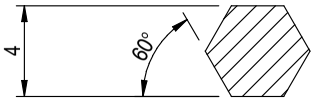
SIZE: A3 (420mm x 297mm)

DRAWING NUMBER: 001

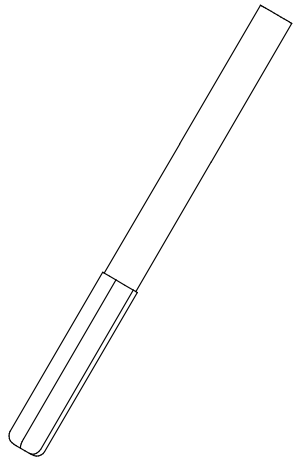
SHEET No.: 1 OF 1

ISSUE: 1





SECTION A-A



DO NOT TAKE MEASUREMENTS FROM DRAWING PRINTS. IF IN DOUBT: ASK.

ISSUE	DATE	DRAWN	APPROVED	DETAILS OF CHANGES

REVISION HISTORY

MATERIAL

PROTECTIVE FINISH

DIMENSIONS IN MM | SCALE AT A3 SIZE:

DRAWN IN ACCORDANCE WITH BS8888

TOLERANCES UNLESS OTHERWISE STATED

LINEAR ± 0.1mm

ANGULAR ± 2deg

SURFACE TEXTURE ✓

REMOVE ALL BURRS AND SHARP EDGES
0.5 MAX. RADIUS OR 45°CHAMFER

UNIVERSITY OF BATH

DEPARTMENT OF MECHANICAL ENGINEERING

ACTIVITY	NAME	DATE
DRAWN	PIA TAUBERT	21/10/12
CHECKED		
APPROVED		

PROJECT: Continuously-variable material properties in RepRap 3D printing

TITLE: Hex Mixer

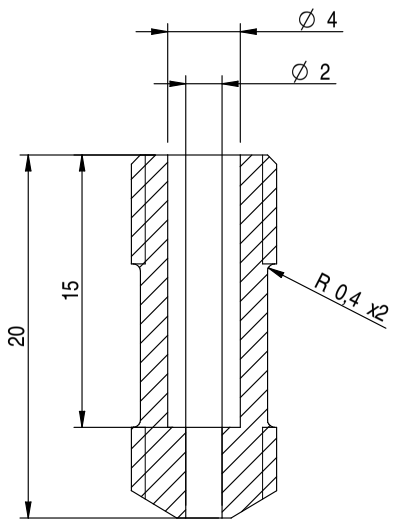
THIRD ANGLE PROJECTION

SIZE A3 420mm x 297mm

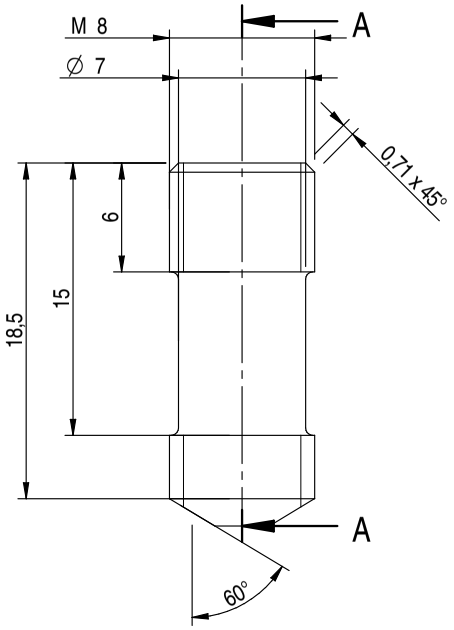
DRAWING NUMBER: 002

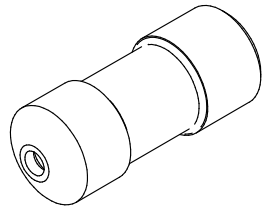
SHEET No.: 1 OF 1

ISSUE: 1



SECTION A-A





DO NOT TAKE MEASUREMENTS FROM DRAWING PRINTS. IF IN DOUBT: ASK.

ISSUE	DATE	DRAWN	APPROVED	DETAILS OF CHANGES

REVISION HISTORY

MATERIAL

PROTECTIVE FINISH

DIMENSIONS IN MM | SCALE AT A3 SIZE:

DRAWN IN ACCORDANCE WITH BS8888

TOLERANCES UNLESS OTHERWISE STATED
LINEAR ± 0.1mm
ANGULAR ± 2deg
SURFACE TEXTURE ✓

REMOVE ALL BURRS AND SHARP EDGES
0.5 MAX. RADIUS OR 45°CHAMFER

UNIVERSITY OF BATH
DEPARTMENT OF MECHANICAL ENGINEERING

ACTIVITY	NAME	DATE
DRAWN	PIA TAUBERT	21/10/12
CHECKED		
APPROVED		

PROJECT: Continuously-variable material properties in RepRap 3D printing

TITLE: Steel Insert 1.75 Hex

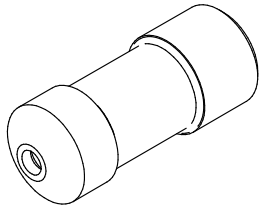
THIRD ANGLE PROJECTION

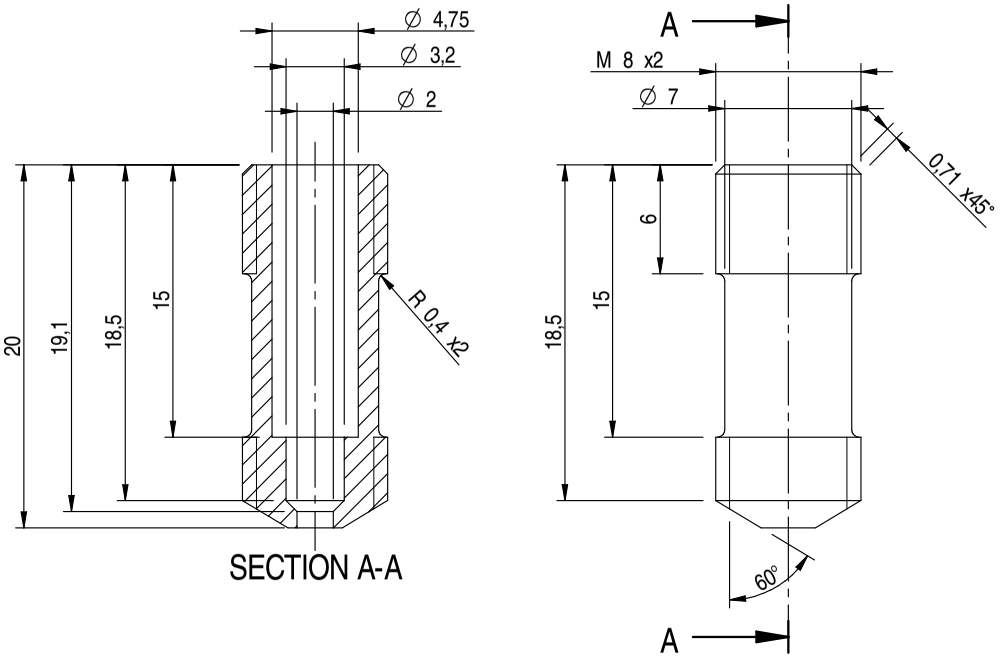
SIZE A3 420mm x 297mm

DRAWING NUMBER: 003

SHEET No.: 1 OF 1

ISSUE: 1





DO NOT TAKE MEASUREMENTS FROM DRAWING PRINTS. IF IN DOUBT: ASK.

ISSUE	DATE	DRAWN	APPROVED	DETAILS OF CHANGES

REVISION HISTORY

MATERIAL

PROTECTIVE FINISH

DIMENSIONS IN MM | SCALE AT A3 SIZE:

DRAWN IN ACCORDANCE WITH BS8888

TOLERANCES UNLESS OTHERWISE STATED
LINEAR ± 0.1mm
ANGULAR ± 2deg
SURFACE TEXTURE ✓

REMOVE ALL BURRS AND SHARP EDGES
0.5 MAX. RADIUS OR 45°CHAMFER

UNIVERSITY OF BATH
DEPARTMENT OF MECHANICAL ENGINEERING

ACTIVITY	NAME	DATE
DRAWN	PIA TAUBERT	21/10/12
CHECKED		
APPROVED		

PROJECT: Continuously-variable material properties in RepRap 3D printing

TITLE: Steel Insert 3.00 Hex

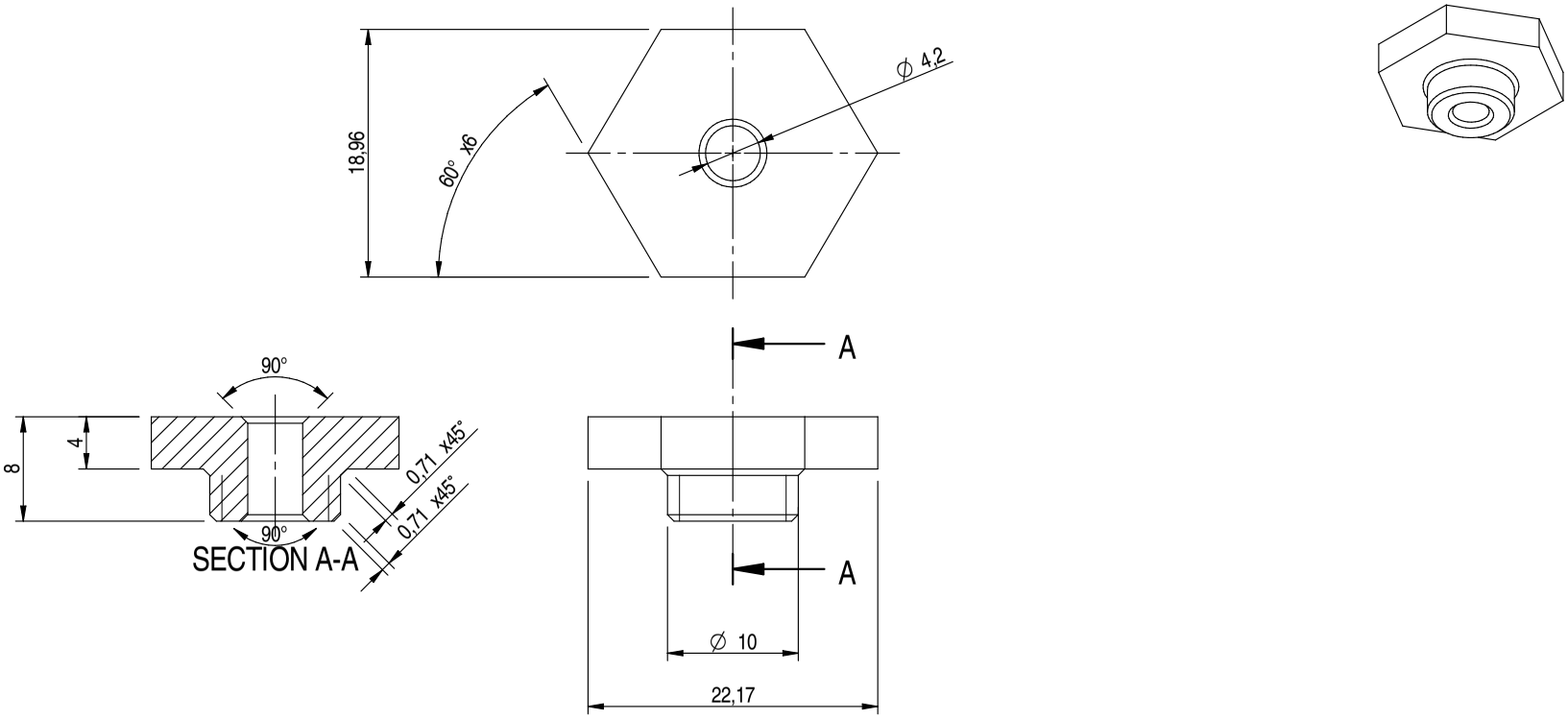
THIRD ANGLE PROJECTION

SIZE A3 420mm x 297mm

DRAWING NUMBER: 004

SHEET No.: 1 OF 1

ISSUE: 1



DO NOT TAKE MEASUREMENTS FROM DRAWING PRINTS. IF IN DOUBT: ASK.

ISSUE	DATE	DRAWN	APPROVED	DETAILS OF CHANGES

REVISION HISTORY

MATERIAL

PROTECTIVE FINISH

DIMENSIONS IN MM | SCALE AT A3 SIZE:

DRAWN IN ACCORDANCE WITH BS8888

TOLERANCES UNLESS OTHERWISE STATED
LINEAR ± 0.1mm
ANGULAR ± 2deg
SURFACE TEXTURE ✓

REMOVE ALL BURRS AND SHARP EDGES
0.5 MAX. RADIUS OR 45°CHAMFER

UNIVERSITY OF BATH
DEPARTMENT OF MECHANICAL ENGINEERING

ACTIVITY	NAME	DATE
DRAWN	PIA TAUBERT	21/10/12
CHECKED		
APPROVED		

PROJECT: Continuously-variable material properties in RepRap 3D printing

TITLE: Seal Bolt

THIRD ANGLE PROJECTION

SIZE A3 420mm x 297mm

DRAWING NUMBER: 005

SHEET No.: 1 OF 1

ISSUE: 1

9.2 C++ programme: Creating G-code

```
 1: #include <iostream>
 2: #include <fstream>
 3: #include <string>
 4: #include <math.h>
 5:
 6: using namespace std;
 7:
 8: int round_value(double number)
 9: {
10:     return int(number +0.5);
11: }
12:
13: double extrude(double x, double y)
14: {
15:     return (pow((0.5/1.75), 2)*sqrt(pow(x, 2)+pow(y, 2)));
16: }
17:
18: int main()
19: {
20:     double length = 0, width = 0, height = 0;
21:     double layer_height = 0.3, nozzle_dia = 0.5*1.4;
22:     double feedrate_fast = 15000.0, feedrate_peri = 500, feedrate_infill = 500;
23:     double e_pos = 0, x_pos = 0, y_pos = 0, x_temp = 0, y_temp = 0;
24:     int nof_perimeter = 0;
25:
26:     double bed_centre = 100;
27:     double x_direction = 1, y_direction = 1; // +1: from min to max;
28:                                              // -1: from max to min
29:
30:     cout << "Enter the length of the object (x): ";
31:     cin >> length;
32:     cout << endl << "Enter the width of the object (y): ";
33:     cin >> width;
34:     cout << endl << "Enter the height of the object (z): ";
35:     cin >> height;
36:
37:
38:     double x_min = bed_centre - length/2, x_max = bed_centre + length/2,
39:            y_min = bed_centre - width/2, y_max = bed_centre + width/2;
40:     double line_thickness = length /
41:            ((round_value(length*1000) / round_value(nozzle_dia*1000)) + 1);
42:
43:     ofstream gcode ("testcube_mod.gcode");
44:
45:     gcode << "M190 S60 \nM104 S200 \nG28 \nM106 \nM109 S200 \nG90 \nG21 " <<
46:              "\nG92 E0 \nM82 \n";
47:     // print variable with x
48:
49:     for (int layer_count = 1; layer_count <= height/layer_height; layer_count++)
50:     {
51:         gcode << "G1 Z" << layer_height*layer_count << " F" << feedrate_fast
52:               << "\n";
53:         if (layer_count == 1)
54:         {
55:            //x_temp = x_min - 4; y_temp = y_min - 4; x_pos += x_temp;
56:            //y_pos += y_temp; e_pos += 2.0; gcode << "G1 X" << x_pos << " Y"
57:            //<< y_pos << " F" << 2*feedrate_infill << " E" << e_pos << "\n";
58:            x_temp = x_min - 4;
```

```
59:             y_temp = y_min - 4;
60:             x_pos += x_temp;
61:             y_pos += y_temp;
62:             gcode << "G1 X" << x_pos << " Y" << y_pos << " F"
63:                   << 0.5*feedrate_fast << "\n";
64:             x_temp = 0.5;
65:             y_temp = 0.5;
66:             x_pos += x_temp;
67:             y_pos += y_temp;
68:             e_pos += 5.0;
69:             gcode << "G1 X" << x_pos << " Y" << y_pos << " F" << feedrate_infill
70:                   << " E" << e_pos << "\n";
71:
72:             //for (int lines = 1; lines <= 4; lines++)
73: //            {                                 // if outer shell is to be printed
74: //                if (x_direction*y_direction == 1)
75: //                {
76: //                     x_temp = x_direction*(length + 8); y_temp = 0;
77: //                     x_pos += x_temp; y_pos += y_temp;
78: //                     e_pos += extrude(x_temp, y_temp);
79: //                     gcode << "G1 X" << x_pos << " Y" << y_pos << " F" <<
80: //                     feedrate_peri << " E" << e_pos << "\n";
81: //                     x_direction *= -1;
82: //                }
83: //                else if (x_direction*y_direction == -1)
84: //                {
85: //                     x_temp = 0; y_temp = y_direction*(width + 8);
86: //                     x_pos += x_temp; y_pos += y_temp;
87: //                     e_pos += extrude(x_temp, y_temp); gcode << "G1 X" <<
88: //                           x_pos << " Y" << y_pos << " E" << e_pos << "\n";
89: //                     y_direction *= -1;
90: //                }
91: //            }
92:             e_pos -= 0.5;
93:             gcode << "G1 F1800.000 E" << e_pos << "\n";
94:             x_temp = 3.5;
95:             y_temp = 3.5;
96:             x_pos += x_temp;
97:             y_pos += y_temp;
98:             gcode << "G1 X" << x_pos << " Y" << y_pos << " F" << feedrate_fast
99:                   << "\n";
100:        }
101:
102:
103:        //e_pos += 1.5; gcode << "G1 F1800.000 E" << e_pos << "\n";
104:
105:        // perimeter
106:        for (int peri = 0; peri < nof_perimeter; peri++)
107:        {
108:            for (int lines = 1; lines <= 4; lines++)
109:            {
110:                if (x_direction*y_direction == 1)
111:                {
112:                    x_temp = x_direction*(length-2*line_thickness*peri);
113:                    y_temp = 0;
114:                    x_pos += x_temp;
115:                    y_pos += y_temp;
116:                    e_pos += extrude(x_temp, y_temp);
```

```
117:                     gcode << "G1 X" << x_pos << " Y" << y_pos << " F"
118:                           << feedrate_peri << " E" << e_pos << "\n";
119:                     x_direction *= -1;
120:                }
121:                else if (x_direction*y_direction == -1)
122:                {
123:                     x_temp = 0;
124:                     y_temp = y_direction*(width-2*line_thickness*peri);
125:                     x_pos += x_temp;
126:                     y_pos += y_temp;
127:                     e_pos += extrude(x_temp, y_temp);
128:                     gcode << "G1 X" << x_pos << " Y" << y_pos << " E" << e_pos
129:                           << "\n";
130:                     y_direction *= -1;
131:                }
132:           }
133:
134:           x_temp = x_direction*line_thickness;
135:           y_temp = y_direction*line_thickness;
136:           x_pos += x_temp;
137:           y_pos += y_temp;
138:           gcode << "G1 X" << x_pos << " Y" << y_pos << " F" << feedrate_fast
139:                 << "\n";
140:      }
141:
142:      // infill
143:      for (int line_count = 0;
144:                line_count <= (length / line_thickness - 2*nof_perimeter);
145:                line_count++)
146:      {
147:           if (line_count != 0)
148:           {
149:                x_temp = x_direction*line_thickness;
150:                y_temp = 0;
151:                x_pos += x_temp;
152:                y_pos += y_temp;
153:                gcode << "G1 X" << x_pos << " Y" << y_pos << " F"
154:                      << feedrate_fast << "\n";
155:           }
156:           x_temp = 0;
157:           y_temp = y_direction*(width-2*line_thickness*nof_perimeter);
158:           x_pos += x_temp; y_pos += y_temp;
159:           e_pos += extrude(x_temp, y_temp);
160:           gcode << "G1 X" << x_pos << " Y" << y_pos << " F"
161:                 << feedrate_infill << " E" << e_pos << "\n";
162:           y_direction *= -1;
163:      }
164:
165:      e_pos -= 0.5;
166:      gcode << "G1 F1800.000 E" << e_pos << "\n";
167:      x_temp = x_direction*nof_perimeter*line_thickness;
168:      y_temp = y_direction*nof_perimeter*line_thickness;
169:      x_pos += x_temp;
170:      y_pos -= y_temp;
171:      gcode << "G1 X" << x_pos << " Y" << y_pos << " F" << feedrate_fast
172:            << "\n";
173:
174:      x_direction *= -1;
```

```
175:     }
176:
177:     x_temp = x_direction*10*line_thickness;
178:     y_temp = y_direction*10*line_thickness;
179:     x_pos -= x_temp; y_pos -= y_temp;
180:     gcode << "G1 X" << x_pos << " Y" << y_pos << " F" << feedrate_fast << "\n";
181:     for (int count = 1; count <= 15; count++)
182:     {
183:         x_temp = x_direction*line_thickness;
184:         y_temp = y_direction*line_thickness;
185:         x_pos -= x_temp; y_pos -= y_temp;
186:         gcode << "G1 X" << x_pos << " Y" << y_pos << " F" << feedrate_fast
187:               << "\n";
188:     }
189:     gcode << "M104 S0 \nM140 S0 \nG28 X0 \nM107 \nM84 \n";
190:     gcode.close();
191:     system("pause");
192:
193:     return 0;
194: }
195:
```

9.3 C++ programme: Modifying G-code

```
 1: #include <iostream>
 2: #include <fstream>
 3: #include <string>
 4: #include <stdlib.h>
 5: #include <math.h>
 6:
 7: #define PI 3.14159265359
 8:
 9: using namespace std;
10:
11: int round_value(double number)
12: {
13:     return int(number +0.5);
14: }
15:
16: double get_value(string line, const char* character)
17: {
18:   int pos;
19:   double value = -1;
20:
21:   pos = line.find(character);
22:
23:   if (pos != string::npos)
24:      value = atof(line.substr(pos+1, line.find(" ", pos)).c_str());
25:
26:   return value;
27: }
28:
29:
30: int main()
31: {
32:   int noflines = 0;
33:   double x_min = 200, x_max = 0, y_min = 200, y_max = 0;
34:
35:   string current_line;
36:
37:   double e_shift_value = 353.4407199/8; //353.4407199;
38:   double x_mod = 0, y_mod = 0;
39:
40:   double old_e_value = 0;
41:   double old_old_e_value = 0;
42:   double old_x_value = 0;
43:   double old_y_value = 0;
44:
45:   ifstream gcode ("testcube_mod.gcode");
46:   if (gcode.is_open())
47:   {
48:      while (!gcode.eof())
49:      {
50:            getline(gcode, current_line);
51:            if ((current_line.find("G1") == 0) &&
52:               (current_line.find("E") != string::npos))
53:            {
54:               double x_value = get_value(current_line, "X");
55:               double y_value = get_value(current_line, "Y");
56:
57:               if ((x_value < x_min) && (x_value > -1))
58:                   x_min = x_value;
```

```
59:                 else if ((x_value > x_max) && (x_value > -1))
60:                     x_max = x_value;
61:
62:                 if ((y_value < y_min) && (y_value > -1))
63:                     y_min = y_value;
64:                 else if ((y_value > y_max) && (y_value > -1))
65:                     y_max = y_value;
66:
67:                 double e_value = get_value(current_line, "E");
68:
69:                 if ((e_value >= e_shift_value) && (round_value(x_mod) == 0) &&
70:                    (round_value(y_mod) == 0))
71:                 {
72:                     x_mod = x_value;
73:                     y_mod = y_value;
74:                     cout << x_mod;
75:                 }
76:             }
77:         }
78:         gcode.clear();
79:         gcode.seekg(gcode.beg);
80:
81:         ofstream gcode_mod ("testcube_mod_soft.gcode");
82:
83:         double A = 0.5;
84:         double B = 0.5;
85:
86:         double A_retract = 0.5;
87:         double B_retract = 0.5;
88:
89:         int choice = 0;
90:
91:         cout << "Secify which function the colour transition shall be of:" << endl;
92:         cout << "1: Single colour (extruder 1 -> blue)" << endl
93:              << "2: Single colour (extruder 2 -> white)" << endl;
94:         cout << "3: Uniform colour of specified ratio" << endl
95:              << "4: Linear transition with x" << endl;
96:         cout << "5: Linear transition with y" << endl
97:              << "5: Linear transition with x and y" << endl;
98:         cout << "7: Sinusoidal with x" << endl << "8: Sinusoidal with y" << endl;
99:         cin >> choice;
100:
101:        while ((choice != 1) && (choice != 2) && (choice != 3) && (choice != 4)
102:               && (choice != 5) && (choice != 6) && (choice != 7) && (choice != 8))
103:        {
104:               cout << "Invalid choice. Please select a valid option." << endl;
105:               cin >> choice;
106:        }
107:
108:        while (!gcode.eof())
109:        {
110:               getline(gcode, current_line);
111:               if ((current_line.find("G1") == 0) &&
112:                  (current_line.find("E") != string::npos))
113:               {
114:                  noflines++;
115:
116:                  double x_value = get_value(current_line, "X");
```

```
117:                double y_value = get_value(current_line, "Y");
118:                double e_value = get_value(current_line, "E");
119:
120:                switch (choice)
121:                {
122:                   case 1: // Single colour (extruder 1)
123:                   {
124:                       if (noflines == 1)
125:                       {
126:                          A = 1;
127:                          B = 0;
128:                       }
129:                       break;
130:                   }
131:                   case 2: // Single colour (extruder 2)
132:                   {
133:                       if (noflines == 1)
134:                       {
135:                          A = 0;
136:                          B = 1;
137:                       }
138:                       break;
139:                   }
140:                   case 3: // Uniform colour of specified ratio
141:                   {
142:                       if (noflines == 1)
143:                       {
144:                          cout << "Specify the required ratio in the form of "
145:                                  "E1:E2 where E1 and E2 are specified in "
146:                                  "percent (e.g. 50:50 -> blue:white)" << endl;
147:                          string ratio = "";
148:                          cin >> ratio;
149:
150:                          bool valid_ratio = false;
151:
152:
153:                          while (valid_ratio == false)
154:                          {
155:
156:                             while (ratio.find(":") == string::npos)
157:                             {
158:                                 cout << "Invalid format. Please reenter the "
159:                                         "ratio in the specified form." << endl;
160:                                 cin >> ratio;
161:                             }
162:
163:                             A = atof(ratio.substr(0, ratio.find(":")).
164:                                 c_str())/100;
165:                             B = atof(ratio.substr(ratio.find(":")+1,
166:                                 ratio.length()).c_str())/100;
167:
168:                             if ((round_value(A+B) == 1) && (A <= 1) && (A >= 0)
169:                                && (B <= 1) && (B >= 0))
170:                                 {valid_ratio = true;}
171:                             else
172:                                 {ratio = "";}
173:                          }
174:                       }
```

```
175:                     break;
176:                 }
177:                 case 4: // Linear transition with x
178:                 {
179:                     if ((x_value > -1) && (y_value > -1))
180:                     {
181:                        if (x_value >= old_x_value)
182:                           if (((x_value + x_mod - x_min) >= x_min) &&
183:                              ((x_value + x_mod - x_min) <= x_max))
184:                              A = (x_value + x_mod - 2*x_min)/(x_max - x_min);
185:                           else
186:                              A = (2*x_max - x_value - x_mod)/(x_max - x_min);
187:                        else
188:                           if (((x_value - x_mod + x_min) >= x_min) &&
189:                              ((x_value - x_mod + x_min) <= x_max))
190:                              A = (x_value - x_mod)/(x_max - x_min);
191:                           else
192:                              A = -(x_value - x_mod)/(x_max - x_min);
193:
194: //                        A = (x_value-x_min)/(x_max-x_min);   // original line
195:                        B = 1 - A;
196:                        A *= pow((1.75/2.6), 2);
197:                     }
198:                     break;
199:                 }
200:                 case 5: // Linear transition with y
201:                 {
202:                     if ((x_value > -1) && (y_value > -1))
203:                     {
204:                        A = (y_value-y_min)/(y_max-y_min);
205:                        B = 1 - A;
206:                     }
207:                     break;
208:                 }
209:                 case 6: // Linear transition with x and y
210:                 {
211:                     if ((x_value > -1) && (y_value > -1))
212:                     {
213:                        A = ((x_value-x_min)/(x_max-x_min))*
214:                            ((y_value-y_min)/(y_max-y_min));
215:                        B = 1 - A;
216:                     }
217:                     break;
218:                 }
219:                 case 7: // Sinusoidal with x
220:                 {
221:                     if ((x_value > -1) && (y_value > -1))
222:                     {
223:                        A = sin(((x_value-x_min)/(x_max-x_min))*PI);
224:                        B = 1 - A;
225:                     }
226:                     break;
227:                 }
228:                 case 8: // Sinusoidal with y
229:                 {
230:                     if ((x_value > -1) && (y_value > -1))
231:                     {
232:                        A = sin(((y_value-y_min)/(y_max-y_min))*PI);
```

```
233:                         B = 1 - A;
234:                     }
235:                     break;
236:                 }
237:                 default: cout << "Invalid choice. An equal colour ratio of "
238:                                  "50:50 will be used." << endl;
239:                     break;
240:             }
241:
242:             if ((e_value > old_e_value)) //&& (e_value > old_old_e_value))
243:                 gcode_mod << current_line << " A" << A << " B" << B << "\n";
244:             else
245:                 gcode_mod << current_line << " A" << A_retract << " B"
246:                           << B_retract << "\n";
247:
248:             old_old_e_value = old_e_value;
249:             old_e_value = e_value;
250:             old_x_value = x_value;
251:             old_y_value = y_value;
252:         }
253:
254:         else
255:             gcode_mod << current_line << "\n";
256:     }
257:     gcode_mod.close();
258:     gcode.close();
259:
260:     cout << "GCode has been modified." << endl;
261:   }
262:   else cout << "File could not be opened.";
263:
264:   system("pause");
265:
266:   return 0;
267: }
268:
```