Implementation, Development and Testing of a Robotic 3D Printing System

MEng Final Project Report

Louis Huygens (199136733)
Integrated Mechanical and Electrical Engineering, Fifth Year
Supervised by Runan Zhang, Assessed by Leen Jabban

Published 08/05/2024

This project has been approved by the University of Bath Ethics board.

Ref: 3547-3428

Word Count (Abstract): 309

Word Count (Body): 11272

Acknowledgements

GenAI was not used in the preparation of this assignment.

There are several people I would like to mention for their help with this project. Firstly, a massive thank you to my supervisor, Dr Runan Zhang, for his help with setting up and programming any equipment in the lab, and for answering any questions I sent his way.

I would also like to thank Ian Pearse, with whom I cooperated on this project. Ian was a pleasure to work with and helped me greatly with the initial set up of the system. His independent results also helped me confirm some of the behaviour I had observed with the robot arm.

Finally, I would like to thank Ben Welch, Aishwara Bhattarai and Zaynab Cadow, who gave up their own personal time to assist with the project and helped lighten the overall load.

Contents

Figures

Equations

Tables

Appendices

Symbols and Acronyms

- 2D – Two dimensional
- 3D – Three Dimensional
- AM – Additive Manufacturing
- AWG – American Wire Gauge
- CAGR – Compound Annual Growth Rate
- CNC – Computer Numerical Control
- DoF – Degree of Freedom
- EM – Electromagnetic
- FDM – Fused Deposition Modelling
- FEA – Finite Element Analysis
- I/O – Input/Output
- PEEK – Polyether Ether Ketone
- PID – Proportional, Integral, Derivative
- PLA – Poly Lactic Acid
- PSU – Power Supply Unit
- PWM – Pulse Width Modulation
- SLA – Stereolithography
- SLS – Selective Laser Sintering
- STL – Stereolithography
- UK – United Kingdom

Abstract

AM (Additive Manufacturing) is revolutionising component manufacture, with an increasing number of industries turning to the method for the generation of parts with internal structure not possible with subtractive methods. This has significantly diversified the field and allowed for improved material efficiencies and strengths. Recently other fields have been gaining interest in the technology, such as the construction and biomed industry.

Multi axis printing is a step up from conventional 3D (Three Dimensional) printing methods, which allows for the orientation of the print head to be altered during the printing process. This allows for the generation of parts with increasingly complex geometries, enabling highly detailed and specific structures and tailoring of the mechanical properties of a part to best suit a specific set of loading requirements.

This report details the development of a robotic 3D printing system, using an ABB IRB 120 6 DoF (Degree of Freedom) robot arm, retrofitted with off the shelf components. The purpose is to produce a system that is fully capable of replicating the performance of a desktop system, that will act as a base for future development of a system capable of multi axis printing. The system is programmed to convert conventional GCode into a path for the arm and perform autonomous bed levelling.

The completed system was able to produce parts with a dimensional accuracy of ±0.46mm, and was able to account for bed variances of ±6mm, while printing at speeds of 50mm/s. Thus, the system was deemed capable of meeting the basic requirements of a desktop printer. Further improvements could be made to improve print quality, with untuned retraction and poor bed adhesion as the major issues.

The system demonstrates the ability to produce a capable robotic 3D printing system relatively cheaply and utilising mostly off the shelf hobby grade components. Further development will help establish a multi axis printing platform.

1. Introduction

AM has developed rapidly over the past decade, with techniques like SLS (selective Laser Sintering), SLA (Stereo Lithography), inkjet and FDM (Fused Deposition Modelling) being the most prevalent [1]. However, despite the improved manufacturing capabilities AM offers, the technology is still limited to printing with sequential layers with conventional methods.

By attaching a print head to the end effector of a 6 DoF robot arm, a multi axis printing system can be achieved. Unlike FDM printing in desktop solutions, where the orientation of the print head is constrained and the system simply moves along the X, Y and Z axis', a robotic solution can fully rotate and orient the print head across 6 axes.

This allows for the deposition of material in any direction, as opposed to building up each layer sequentially from the build plate. Use of a robotic arm also provides a significantly larger and more versatile print area than desktop printers, due to the ability of 6 DoF arms to rotate 360° about their base.

In this project, the development and testing of a complete robotic 3D printing system is outlined. The introduction to this report discusses the relevant background information, the applications of the technology, and summarises the scope and aims of the project. This is followed by the methodology used to achieve these aims, and the results of tests used to gauge system performance. A critical analysis of the results is provided, as well as an analysis of the overall project.

1.1. Project Scope

1.1.1. Aims

The aim of this project is to produce an affordable robotic 3D printing system, based on the ABB IRB 120 robotic arm [2], with the capability for future development of multi axis printing. The system will utilise off the shelf components and custom-built code to avoid reliance on proprietary software where possible. This includes an E3D Hemera print head [3], using a DM856 stepper driver to control the feed motor [4]. Temperature of the hot end will be controlled by a print head control unit, constructed by university technicians.

The system will be designed to meet or exceed the capabilities of a desktop FDM printer. Whilst the machine will have the ability to perform multi axis printing, this will not be considered part of the scope of this project, and instead a subject for future development.

The system will also incorporate some form of automated bed levelling. This will be done through the implementation of a BLTouch bed levelling sensor [5]. The system should be able to account for an acceptable variance in bed level height. Demonstration of printing on non-planar surfaces is again not within the scope of this project.

A Gantt chart showing the proposed timeline for the project can be found in Appendix 1. The project is broken down into 3 phases representing key milestones that need to be hit: Mechanical Integration, Programming and Testing, and Bed Levelling Integration.

1.1.2. Targets

A full breakdown of the target for this project can be found in the project scoping and planning report [6]. Table 1 gives an overview of the targets and the measurable objectives.

Ref	Targets	Sub Ref	Description	Tolerance
1	System holds all equipment for 3D printing	1.A	All components are mounted to the machine. Cables running to components. Print bed set up in front of machine. [7]	±1mm
		1.B	System retains full range of motion. [2]	Joints maintain rotation of ±180°
		1.C	Bed height and level fully adjustable. [8]	Corner height adjusted ±5mm
2	Extruder capable of extruding filament on demand	2.A	Extruder can reach and maintaining an acceptable temperature. [9] [10]	±5°C
		2.B	Extruder can extrude a demanded length of filament. [11] [12]	±0.5mm
3	Printer capable of completing a series of test prints	3.A	Printer can match print and extrusion speed. [13]	Line width 100% to 120% of the nozzle's diameter
		3.B	Printer can complete a series of test prints to demonstrate dimensional accuracy. [14]	±0.5mm
4	System capable of performing autonomous bed levelling	4.A	Printer can take a single bed height measurement. [15]	±0.06mm
		4.B	Several measurements used to autonomously correct for bed level. [16]	±0.1mm

Table 1 - Targets and objectives of the project.

All objectives have been based off the performance of existing desktop printers, or the performance off a similar 6 DoF arm, and have been cited where necessary. Full details of the testing procedure to obtain

these results is given in the section 2 and 3. Table 2 gives an overview of the tests that have been performed.

Ref	Sub Ref	Tolerance	Testing method
1	1.A	±1mm	Measurement of all components on machine. Pose test to check for interference through full range of motion
	1.B	Joints maintain rotation of ±180°	Pose test to check for full range of motion
	1.C	Corner height adjusted ±5mm	Install print bed. Check degree of level with 3D scanner
2	2.A	±5°C over one hour	Measure and record print head temperature through range of scenarios: idle, extruding, fan on, fan off
	2.B	±0.5mm	E-Step calibration
3	3.A	Line width 100% to 120% of the nozzle's diameter	Line test, performed at various speeds. Measurement and comparison of line widths produced.
	3.B	±0.5mm	Square, Calibration cube and Benchy tests. Comparison of dimension in X, Y and Z
4	4.A	±0.06mm	Comparison of measured values to 3D scanner results
	4.B	±0.1mm	Full print surface test at target bed angle

Table 2 - Testing procedure for each target.

1.2. Background Theory

This section provides background information on 3D printing and robot control, which will be used throughout the report. This includes a basic description of FDM printing technology, how slicers are used, what bed levelling is and how it works, and how robots are controlled. Any governing mathematical principles have been identified.

1.2.1. FDM 3D Printing

FDM is the most common method for desktop 3D printing, since it is the simplest and cheapest to implement. It involves extruding a 1.75mm diameter plastic filament in layers, which are built up on top of each other to create a 3D object [17]. PLA (Poly Lactic Acid) is the most used filament because its non-toxic, biodegradable and easy to print with [18]. Modern printers typically either use cartesian or coreXY kinematics.

Cartesian printers feature a single motor controlling each axis, as shown in Figure 1. This makes them very easy to control and tune, since the number of motor steps directly relates to a change in position in the corresponding axis. These printers typically suffer from vibration due to their moving print bed [19], and as such can rarely achieve speeds over 100mm/s with an accuracy of ±0.5mm [20] [14] . Typical build volumes are around $230mm^3$, with some printers going up to $1m^3$ [21].

Figure 1 - Ender 3 cartesian 3D printer [20].

CoreXY printers use a single motor to control the z axis, and a combination of two motors to control movement in the X and Y axis', shown in Figure 2. This results in more complex kinematics, however if the system is well tuned, it produces much less vibration, and is able to achieve print speeds of up to 500mm/s with a precision of ±0.25mm [22] [23].

Figure 2 - P1S coreXY 3D printer [22].

A printer's extruder is comprised of a hot zone and a cool zone, as shown in Figure 3. The hot zone is where the filament is melted and extruded out the nozzle. Temperature is kept constant in this region via closed loop PID (Proportional, Integral, Derivative) control between the thermistor and the heater cartridge [24]. A cool zone is maintained above the hot zone to prevent heat creep, where molten filament travels up into the body of the extruder and solidifies, causing blockages [25]. The filament is retracted when extrusion is not required to void the hot zone of material and prevent any leakage out of the nozzle.

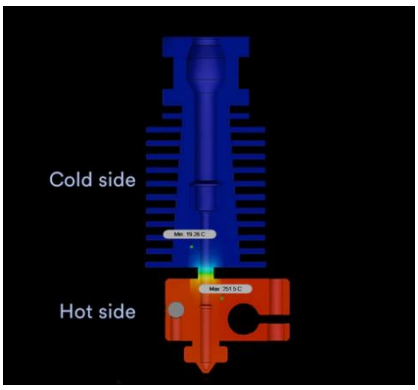


Figure 3 - Depiction of heat zones on an extruder [26].

Strength in 3D printed components is non isotropic and can be reduced by as much as 50% to 75% across the Z axis direction [27]. This is due to the separation between layers acting as an initial point of failure and can result in fatigue crack propagation [28]. Print strength is also highly dependent on material, with specialist materials like PEEK (Polyether Ether Ketone) displaying tensile strengths twice as high as PLA [29]. Print parameters, like speed, temperature, acceleration and nozzle diameter can all influence the overall print quality and component strength [30].

1.2.2. Slicers

3D printers take commands through a control script known as GCode, which is standardised under BS ISO 6983-1:2009 [31]. A slicer is used to convert a 3D STL (Stereolithography) file into GCode [32]. This is done by splitting the 3D model into layers of the target layer height, and writing a series of commands that path a route for the toolhead to effectively cover the full area of each cross section [33]. STL files define the outside surface of a 3D object as a mesh of tessellated triangles [34]. Figure 4 shows an STL of a sphere, with one half using a higher resolution mesh than the other.

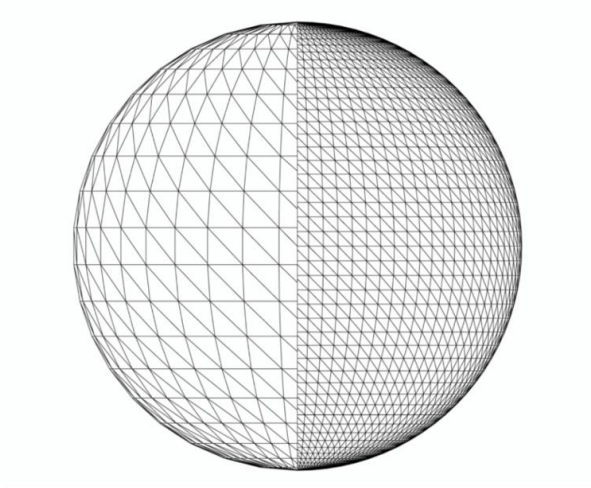


Figure 4 - STL surface of a sphere, using different mesh resolutions [35].

GCode for 3D printers typically consists of 2 commands: G and M. G commands relate to movement of the print head, with G1 indicating a linear movement [36]. This will be preceded by a set of X and Y coordinates, or a Z coordinate if a layer change occurs. E coordinates indicate how far the extruder motor should move during the execution of the G1 movement. If the G1 command is preceded by an F, the print head movement speed will be set to the following number. M commands control additional I/O (Input/Output) on the printer that are not responsible for movement, such as setting temperature and controlling fans.

Since all movements are linear, different slicers handle geometry in different ways, and can use more points to define curves than others. As such, even for the same machine, different results can be achieved simply by using another slicer, even with the same printing parameters [37].

1.2.3. Bed Levelling

Automated bed levelling is used in 3D printers to account for variance in bed height. This can either be due to a small tilt of the bed, or defects in the print bed itself. This is done by probing the bed at a series of points and generating a mesh of height offsets, which is locally applied at each coordinate to keep the relative distance between the nozzle and the print bed the same [38].

Heights in between the probe points are calculated using bilinear interpolation [39]. This takes the heights at 4 corners of a quadrant and infers the relative height of a search point based on the variance across the corners. The generalised equation for a point within the quadrant is given in Equation 1, where $f_{(x, y)}$ represents the height at the point (X, Y) and a, b, c, d are arbitrary constants.

$$f_{(x,y)} = a + bx + cy + dxy$$

Equation 1 - Generalised equation for a point within a quadrant.

Where a quadrant is laid out according to Figure 5.

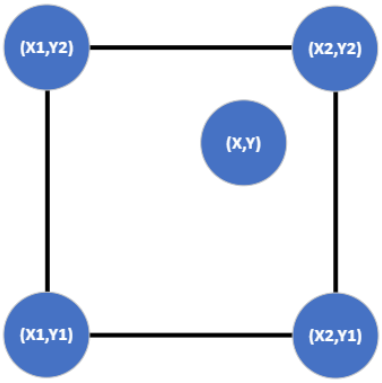


Figure 5 - Layout of local quadrant.

Therefore, the equations of each of the corner points can be written as in Equation 2.

$$f_{(11)} = a + bx_1 + cy_1 + dx_1y_1$$

$$f_{(12)} = a + bx_1 + cy_2 + dx_1y_2$$

$$f_{(22)} = a + bx_2 + cy_2 + dx_2y_2$$

$$f_{(21)} = a + bx_2 + cy_1 + dx_2y_1$$

Equation 2 - Generalised equations for each of the corner points.

Since the coordinates and heights have been recorded at each of these points during the probing process, the equations can be solved simultaneously to find the values of constants a, b, c, d. Hence, the generalised expression in Equation 1 can be realised, and the height at any coordinate within the quadrant can be calculated.

1.2.4. Robot Control

Robot controllers will typically use their own coding languages to generate command programs; in the case of the ABB IRB 120, Rapid is used [2] [40]. This Language uses a set of targets to define a tools position, orientation and robot configuration in space, with commands used to designate movement to each target at a given input speed, with reference to a specific tool and work object. A tool is designated by an end effector position and orientation in space, as well as having mass properties to account for inertia. A work object defines the coordinate system that the robot will use when moving to a specific target and is usually set to the bottom left-hand corner of the build plate.

The robot controller takes this input and converts it into a set of joint angles and speeds required to reach a certain position, following a linear path [41]. This is done by a process called inverse kinematics [42]. Since there will be multiple joint solutions to a given position, the configuration of the robot will need to be predefined as part of the target.

In some configurations, the robot may approach what is known as a singularity, shown in Figure 6. This is where the joint angle approaches a region where there are near infinite solutions, resulting in computational error [43]. This is a result of the determinant of the Jacobian matrix, which is used to calculate the speed and position of each joint, approaching zero, and hence its inverse approaching infinity [44]. There are multiple ways in which a singularity may be avoided, largely involving removing the use of the Jacobian matrix as part of the inverse kinematic solution [45].

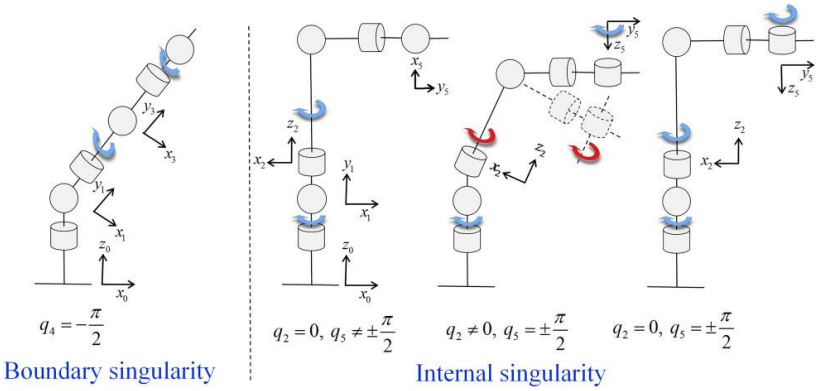


Figure 6 - Singularity conditions for 6 DoF robotic arm [46].

1.3. Implementations

A case study for three potential sectors that could benefit from robotic 3D printing have been identified. These are the construction, bioprinting and component manufacturing industries. For each case, a detailed analysis of the problem 3D printing could solve has been outlined, in comparison to existing technologies.

1.3.1. Construction Industry

The construction industry consumes 48% of the global energy supply [47], and in the UK (United Kingdom) £16.8 billion is paid out for illness or injury each year [48]. The industry struggles to keep up with demand, with only 233,000 houses built in the UK in 2022 for a target of 300,000 [49]. Utilisation of robotic 3D printing for house construction is an emerging solution to this problem. Construction 3D printers require less labour to set up and operate, and generate less waste, reducing the cost of producing a single house by 35%, assuming 10% profit, and an average house price of £282,338 as of 2015 [50]. The outer structure of houses can be constructed in as little as 24 hours, compared to months [51].

There are two main methods currently used, arm based multi axis printers [52] and large-scale gantry printers [53]. Table 3 shows a breakdown of each method [51, 54]. A typical arm printer has a build volume of a 2.75m^3 sphere, compared to around 14.6m X 50.52m X 8.14m on the largest gantry systems, shown in Figure 7 [54]. Dimensional accuracy is not particularly important in either system, since the extrusion width is significantly large that variations measured in mm would be unnoticeable.

Gantry Printers		**Arm Based Printers**	
Pros	**Cons**	**Pros**	**Cons**
• Capable of much larger scale • Very easy to control	• Hard to transport • Requires clearance either side of building • Limited to sequential layering	• Easy to transport and install • Capable of generating more complex structures	• Limited build volume • Hard to program

Table 3 - Breakdown of the pros and cons of gantry vs arm-based construction 3D printers.

Figure 7 - Large scale gantry construction 3D printer [54].

The versatility of arm-based systems allows for much more creative designs, which have been utilised for art installations [55]. These can also be used to create natural inspired structures which have improved mechanical properties for less material usage [56].

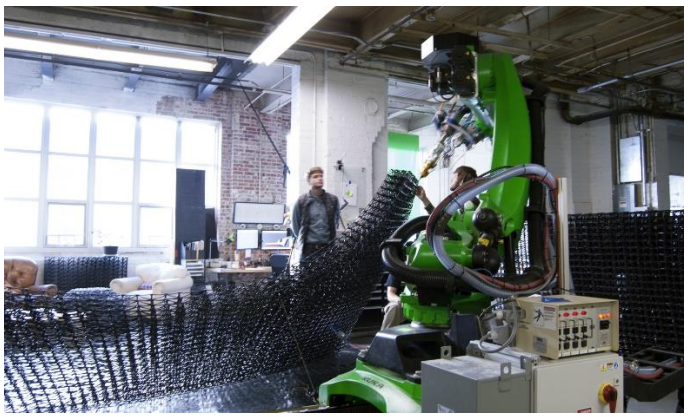


Figure 8 - Arm based printing of pieces for an art installation [55].

1.3.2. Bio Printing

In the UK during 2023, around 11,000 patients were recorded either on the waiting list or suspended waiting list for an organ donation [57]. With a limited amount of organ donors, and compatibility issues, only 1 in 100 deaths result in organs viable of transplantation. 1429 deceased and 958 living donors were recorded in the same year [57]. This means there is significantly more demand for organ donation than there is availability, at a ratio of around 5 to 1.

Organ printing is an emerging solution to this problem. This allows for the creation of completely new organs from cell material. The process builds a scaffold in the desired shape, onto which various bioinks are deposited which contain the cells required. The organ is then allowed to grow on the scaffold, until is complete and ready for transplantation [58]. These structures are made up of multiple cell types, and as such systems need to be capable of depositing multiple materials concurrently [59]. The resulting structure also needs to have a high degree of vascularisation, such that printed cells can receive enough blood flow for long term survival [60]. Figure 9 shows an example of bioink being printed onto a 3D printed scaffold.

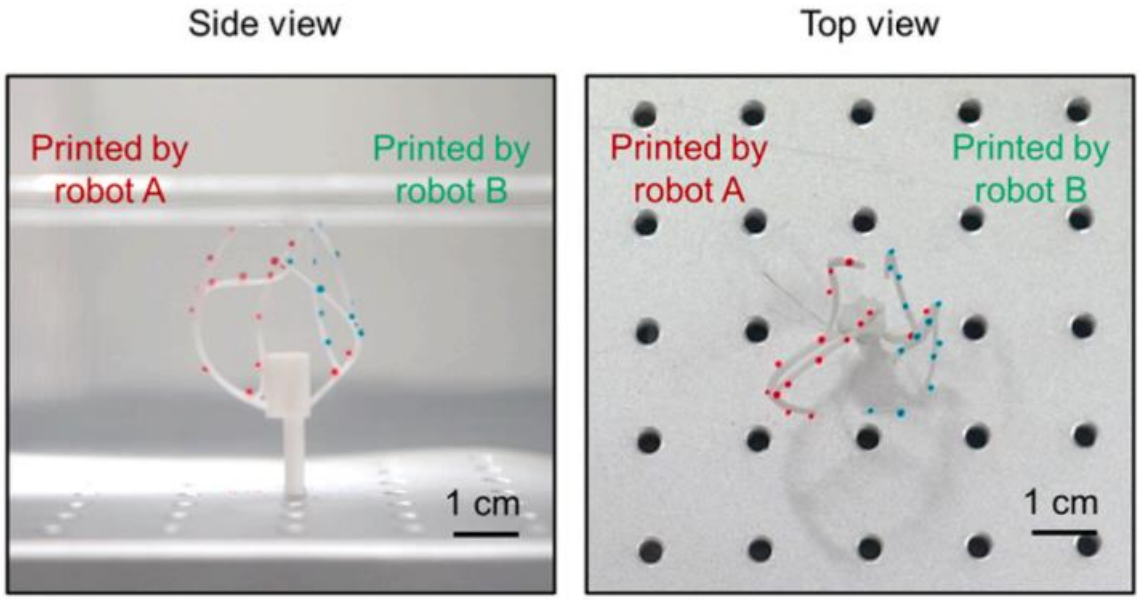


Figure 9 - Printing of different bioinks onto a printed scaffold [60].

These scaffolds have a highly detailed structure, with the whole part being only 3cm across. It features several overhangs and fine points which would be difficult to replicate with conventional printing methods, requiring a dimensional accuracy of around ±0.1mm [60].

Bioprinting can also be applied to facilitate bone tissue repair from fractures, breaks and defects. This is done in a similar method, utilising a scaffold impregnated with cellular material, which is then transplanted onto the damaged site [61]. The bone then grows onto the scaffold structure, reducing the time to repair [62]. The key feature of the scaffold is its porosity, and not its overall dimensional accuracy. Different types of bone tissues have different porosities, with trabecular bone ranging from 40% to 95% porosity [63]. Multi axis printing can allow for the tailoring of porosity and structure throughout the scaffold to better fit the patient.

Methods for in situ organ and tissue repair are also being developed [64]. This is useful for applications where it is impossible to remove the damaged region, such as in muscle, tendons or cartilage. Bioink is injected directly into the damaged site, where cells fill the gap and fully heal the region within 12 weeks, as shown in Figure 10. The method is suitable for damage in cartilage of up to 4cm^2. Multi axis is utilised to reach the site of damage regardless of its orientation, since it may not be possible to properly orient the site in situ.

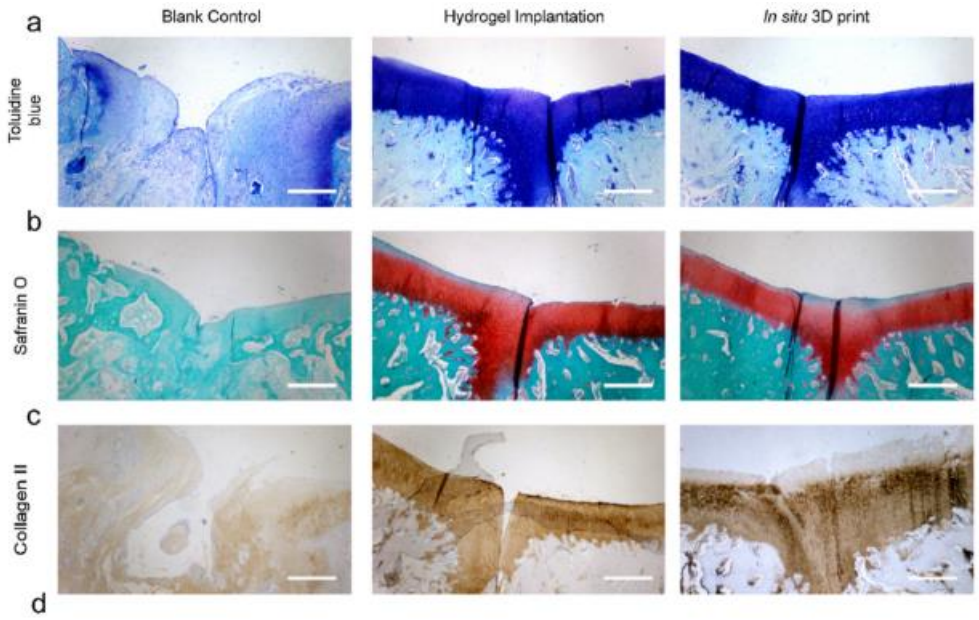


Figure 10 - Comparison with 3D printed cartilage repair to hydrogel implantation [64].

1.3.3. Advanced material properties

Multi axis 3D printing can be used to improve both the mechanical and physical properties of 3D printed components [65]. The printing process typically leaves quite a poor surface finish with artifacts like stair stepping. Multi axis printing approaches can be used to skin over the outer surface of a model, making the outer surface one homogenous layer, as shown in Figure 11 [66]. This is done using a specialist slicer, which calculates the toolpath required whilst avoiding clearance issues [67].

Figure 11 - Comparison of planar slicing with non-planar slicing on stair stepping effect [67].

A multi axis approach can also be used to tackle steep overhangs on components. By varying the print angle during the process, it can be assured that the angle between the print head and the rest of the component does not exceed a critical value. This removes the need for support material, resulting in homogenous models that contour around regions of overhangs, as shown in Figure 12 [68]. By removing support material, the total amount of material required for the print can be reduced, as well as the time to print. It also reduces the amount of post processing required. This can also reduce cross sectional variance in complex models, resulting in a dimensional accuracy of around ±0.4mm [69].

Figure 12 - Demonstration of non-planar printing for support minimisation [68].

Since prints are strongest along lines parallel to the direction of layers, the strength properties of a component can be varied by changing the direction of layer lines [70]. This can allow for the design of materially optimised components, which minimise weight for a given set of loading requirements, using a similar structure to cortical bone, shown in Figure 13 [71]. This is done using an FEA (Finite Element Analysis) analysis of a component to generate a toolpath which better maximises the layer orientation. This increases the part strength for up to 54.6% for a given amount of material [71].

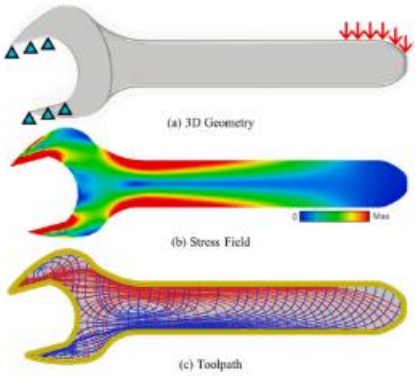


Figure 13 - FEA analysis of a component and the generated toolpath to maximise strength [71].

1.4. Comparisons

Each case has been critically assessed and compared against each other in the context of robotic 3D printing. Comparisons have been made between the scale and precision of systems, their overall marketability and their impact to their respective fields.

1.4.1. Scale and Precision

The construction industry demands the largest scales of any application, with gantry-based machines being larger than the buildings they are constructing. Application of multi axis systems demands the largest arms available, with the potential demand for even larger arms to meet size requirements. Due to the large form factor, the precision of the machine becomes almost irrelevant; when the diameter of the nozzle is measured in hundreds of millimetres, a poor dimensional accuracy of ±10mm will have very little effect on the overall structure of the house. As such, print speed can be maximised, which is one of the key limitations within the industry.

Conversely bioprinting requires the generation of small, highly complex structures. As such, prints speeds will need to be as slow as possible to achieve the demanded ±0.1mm tolerance. This, compounded with the allowed time for organ growth, results in a very slow process. This also requires a highly precise robotic arm and extruder system, likely resulting in a very high startup cost.

The demands for component manufacture are largely dictated by the industry the technology is used in, but for most purposes, the achieved ±0.4mm accuracy and form factors with conventional arms are more than sufficient. Pieces that require a higher tolerance may require more expensive, bespoke systems, however it is more than likely that in these applications additional post processing steps are already in place.

1.4.2. Marketability

With supply of houses only meeting 78% of the demand in the UK, the quick production of low-cost housing is essential. Margins on AM houses are assumed to be relatively low, at only 10% of the unit price. However, the application does have the benefit of reduced labour and material losses, as well as the ability to quickly transition between multiple builds on one site. As such, for this business model, speed of manufacture is essential.

For bioprinting, the business model will largely be oriented around selling complete printing systems, as opposed to selling organs as the product. Given the ratio of demand to supply of organ donors, its sensible to assume a successful implementation of technology would be used worldwide. As such this technology has huge market potential. Given the increasing number of desktop bioprinters hitting the market, it's clear to see that this sector is highly competitive, and being first to market will be a huge benefit to any system manufacturer.

It's hard to gauge the market potential for multi axis AM; the AM industry was valued at $14.5 billion in 2022, with a CAGR (Compound Annual Growth Rate) of 21.6% [72]. CNC (Computer Numerical Control) machining, a similar, more mature field, was valued at $154 billion in the same year [73]. As such, it can be assumed that the market has huge potential to grow. However, growth will largely be driven by the uptake of new systems within industry, a slow process given that the average age of industrial machinery ranges from 15 to 34 years [74]. As such, it may take a significant amount of time for the field to develop enough for smaller businesses to be viable within the space.

1.4.3. Impact to Field

With gantry systems already being successfully demonstrated, multi axis systems provide little improvement to the established process. The systems are more compact and can achieve higher detail, making them more suitable for smaller construction sites. Further development and utilisation of multi axis systems within a construction environment, such as using them to build custom cable, cooling and heating routes directly into the walls of houses, may improve their viability within the space.

Biomedicine is a highly competitive space, and bioprinting is not the only viable method for organ replacement being developed. Success in this field is highly dependent on being first to market or offering a solution at a competitive price point.

Use of multi axis AM in component manufacture has the potential to completely revolutionise the way parts are made, allowing for the construction of parts that would not have been previously possible. However, with new developments, it takes time for designers to be able to utilise the technology to its full capability. This, coupled with the slow turnover of machinery within industry, means it is unlikely that multi axis additive sees significant use in industry in the next 20 years.

2. Methods and Results

The final system integrated all equipment onto the robotic arm, with considerations made to prevent damage and allow for wiring. The system features an independent print head controller, which communicates with the robot controller to synchronise movement and extrusion speed. Programs were made to convert 3D files into toolpaths. This was used to generate a series of test prints, which gauged the overall effectiveness of the system. Figure 14 shows the completed system.

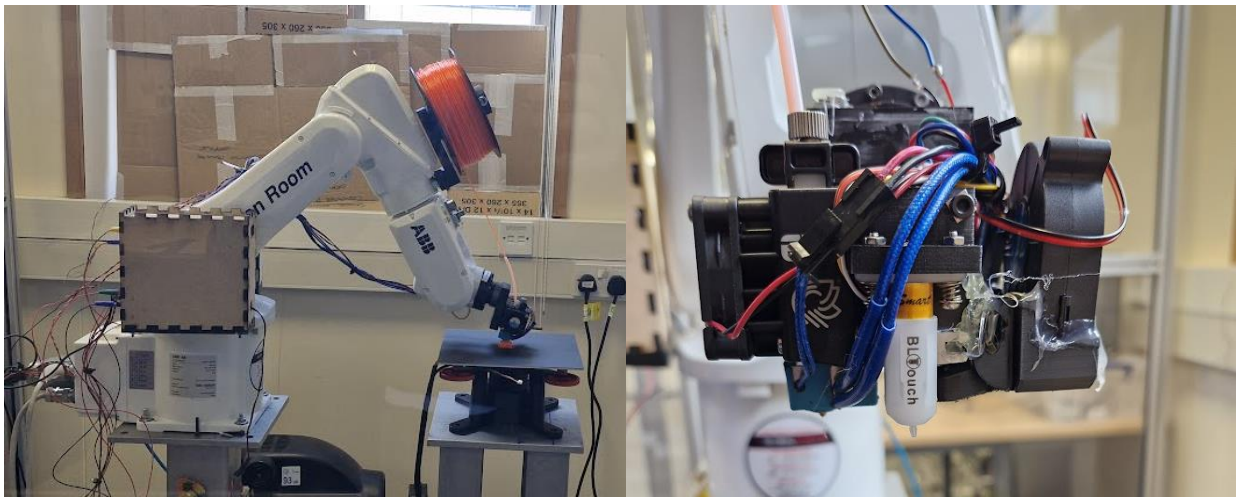


Figure 14 - Images of the completed robotic 3D printing system.

This section details the methods used, with all processes and equations explained. Any testing performed has been detailed, with the results clearly laid out.

2.1. Mechanical Integration

As the Mechanical lead, Ian Pearse was responsible for designing mounts and fitting all components onto the system [75]. This included the extruder, bed level sensor, filament holder, PSU (Power Supply Unit) and stepper driver. Components were mounted in designated safe areas on the robot, as specified in the manual [2]. However, the size of components made this impossible, as demonstrated in Figure 15 and Figure 16, with the PSU exceeding the safe zone by 75mm, and the filament reel completely outside the upper safe zone.

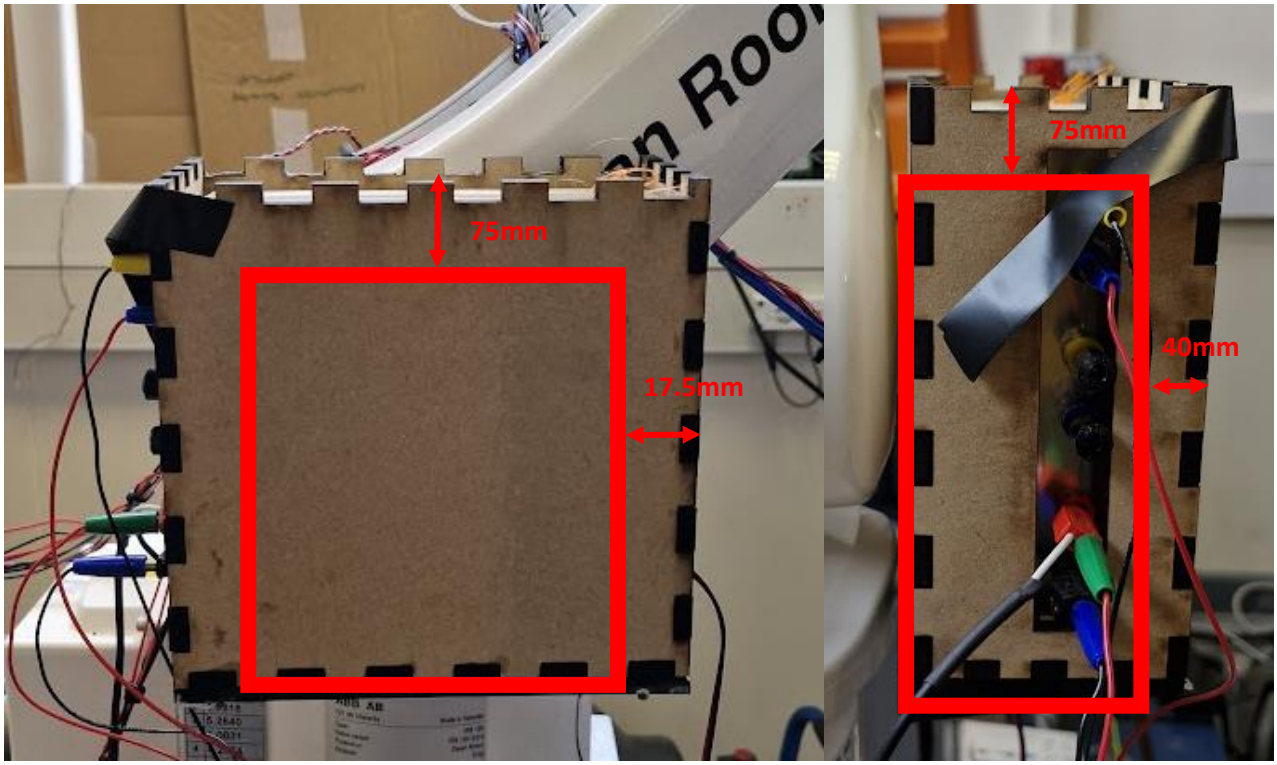


Figure 15 - Fit of components within the robot's lower safe zone.

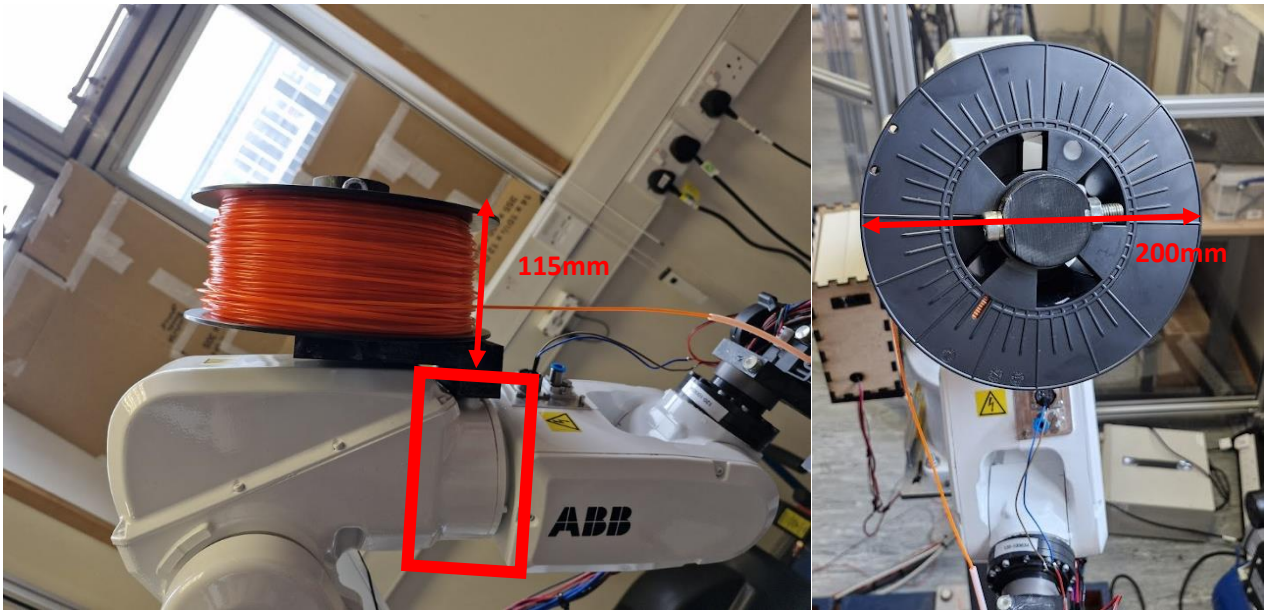


Figure 16 - Fit of components in the robot's upper safe zone.

A full wiring diagram for the system can be seen in Figure 17.

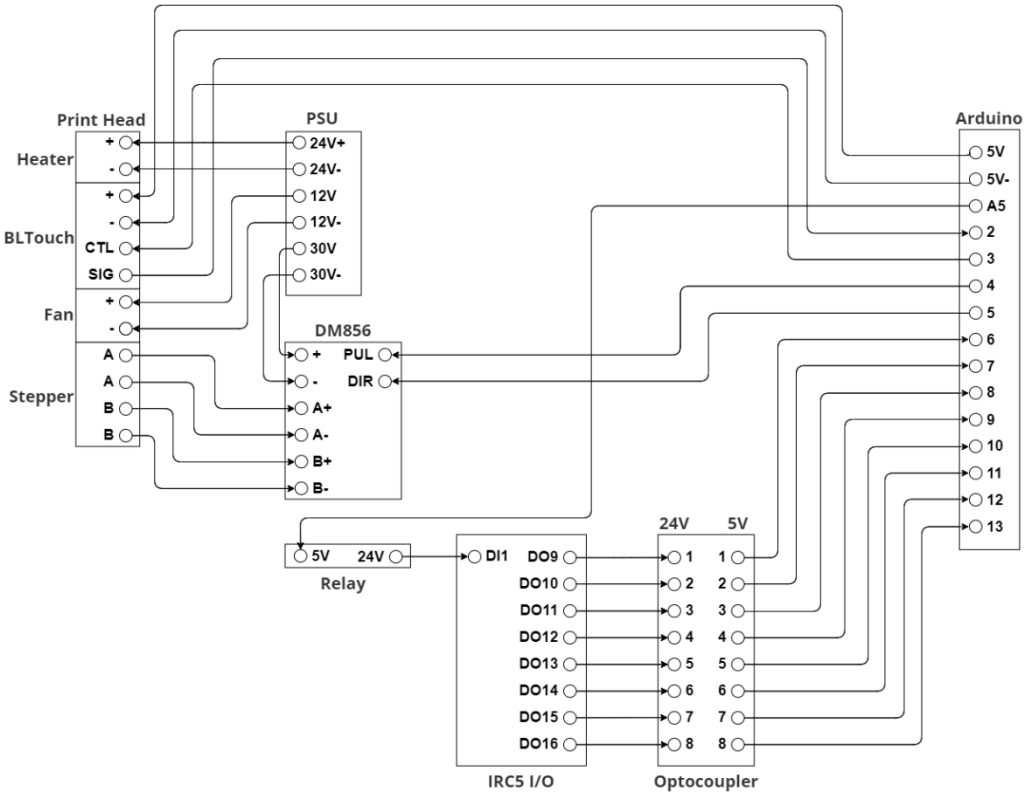


Figure 17 - System circuit diagram.

22AWG (American Wire Gauge) hook up wire was used for all electrical connections, since they were all low power, low voltage, and within requirements [76]. A total of 34 connections were made, four of which were over 5m long reaching between the end effector of the arm and the Arduino stationed at the robot controller. Individual wires were bundled together where applicable to create a wiring harness, set out in BS9550: 1989 [77]. Ties were placed on the harness every 50mm to keep the bundle together, as seen in Figure 18 [78].

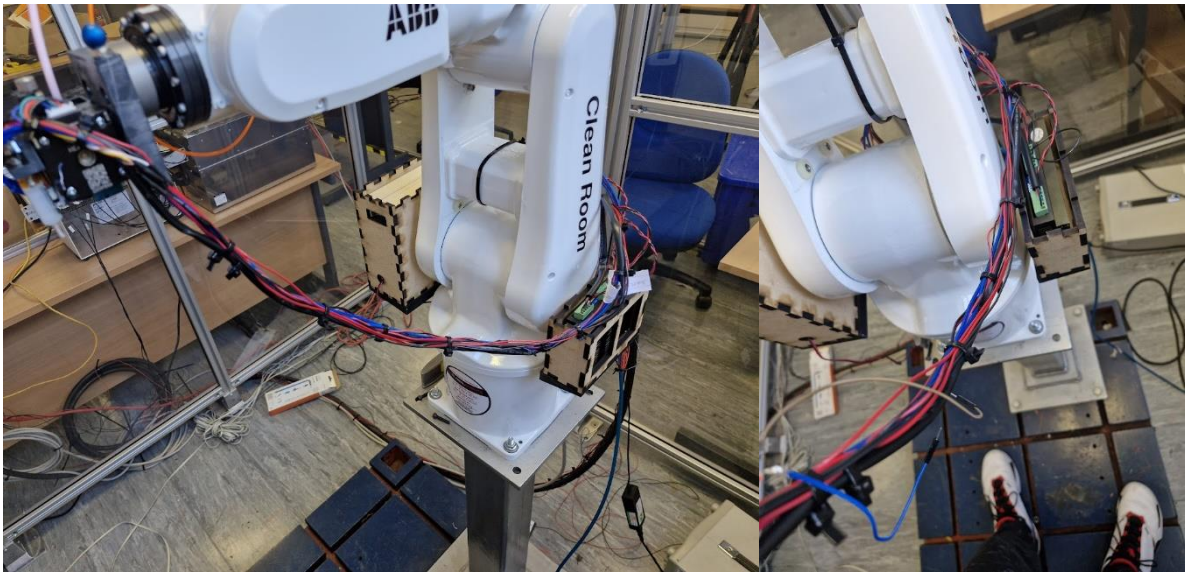


Figure 18 - Bundling of cables to form a wiring harness.

Strain relief was required in the wiring harness to prevent any pinching between moving parts, or bending beyond its maximum bend radius, especially at wire terminals. Cable runs were measured with the robot at full extension and multiplied by 1.5 as a safety factor. This ensured that all wire bend radii were above the minimum of 10mm, especially at the end terminals, where a pull-out force of less than 22N is required for a stable connection [78]. The harness was attached to the robot in key stationary areas, keeping it away from moving joints to prevent pinching.

The robot was able to reach a series of points within a 1m X 1m grid within the designated print area directly in front of the system, without any interference or significant strain on the cables. It was also able to orient the end effector through a series of poses which demonstrate full rotation of the joints. Figure 19 shows the robot during the pose test. The print head was kept at an angle, which help avoid singularities through a straight wrist position. Further details on the range of motion testing can be found in Ian Pearse's final report [75].

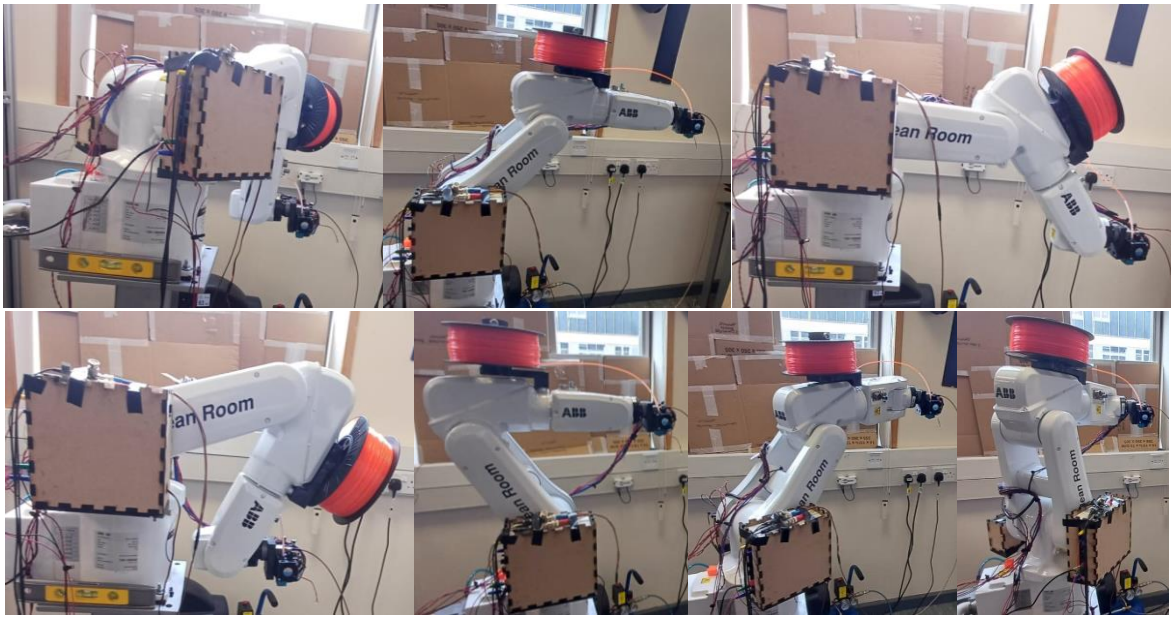


Figure 19 - Robot pose test.

The print bed for the system uses an Ender 3 Y axis gantry, mounted to the table in front of the system. Since the mounting patterns for the table and gantry were different, a mounting adaptor had to be designed and 3D printed, as shown in Figure 20. This part was designed by Ben Welch.

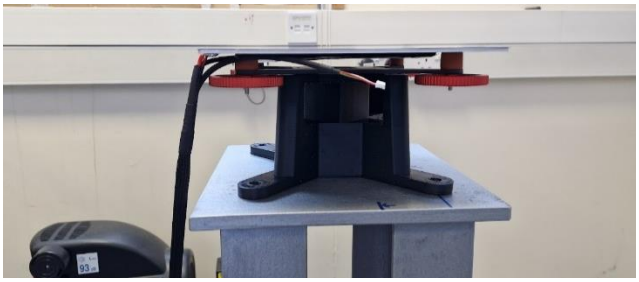


Figure 20 - Print bed mounted to table using a custom designed adaptor.

The gantry uses rubber bungs on each of the four corners of the bed, which can be individually adjusted by tightening the corresponding bed levelling nut. These provide 10mm of travel at each corner, allowing for full adjustment of the level of the table.

2.2. Print Head Control

The print head has two main points of control: temperature and speed. For the system, temperature is managed by a custom control unit built by lab technicians. The extrusion speed is dictated by a feeding stepper motor. Effective control of both factors is required for consistent filament deposition, and hence higher print quality.

2.2.1. Temperature Control

Temperature is controlled by the PSU, which has a built-in bang-bang controller [79]. This reads the temperature value from the hot end's thermistor, and switches power to the heater cartridge based on if the temperature is above or below the set target.

Flow rate of print material from the nozzle is governed by both the materials properties and the temperature of the nozzle, and as such temperature fluctuations can result in print aberrations [80]. To assess whether the control implementation was adequate, temperature measurements were taken. This was done by heating the hot end from cold to the target temperature of 200°C, and measuring the temperature over a 160s period, giving sufficient time for the system to settle. The part cooling fan was turned on for the purposes of testing, to better represent actual print conditions. Measurements were also taken with the hot end at steady state over 30s.

Measurements were taken via the hot end's thermistor, using an Arduino program to read and log the temperature values [81]. The program works by comparing the voltage across a known resistor value to the voltage across the thermistor for a known 5V supply. Then, using the beta coefficient of the thermistor, the temperature can be calculated from its resistance value. Figure 21 shows the comparison circuit used, where the trailing leads indicated were connected across the thermistor. The full Arduino program can be found in Appendix 2.

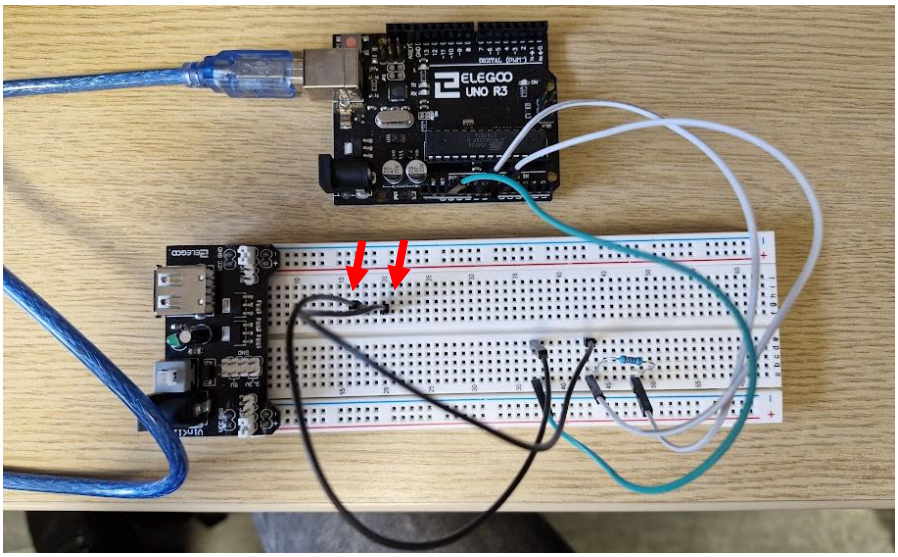


Figure 21 - Comparison circuit used to measure thermistor resistance for temperature calculation.

Results from the test were exported into MATLAB and plotted, as shown in Figure 22. The plot is very noisy, likely due to background interference from other EM (Electromagnetic) sources, such as the system fans.

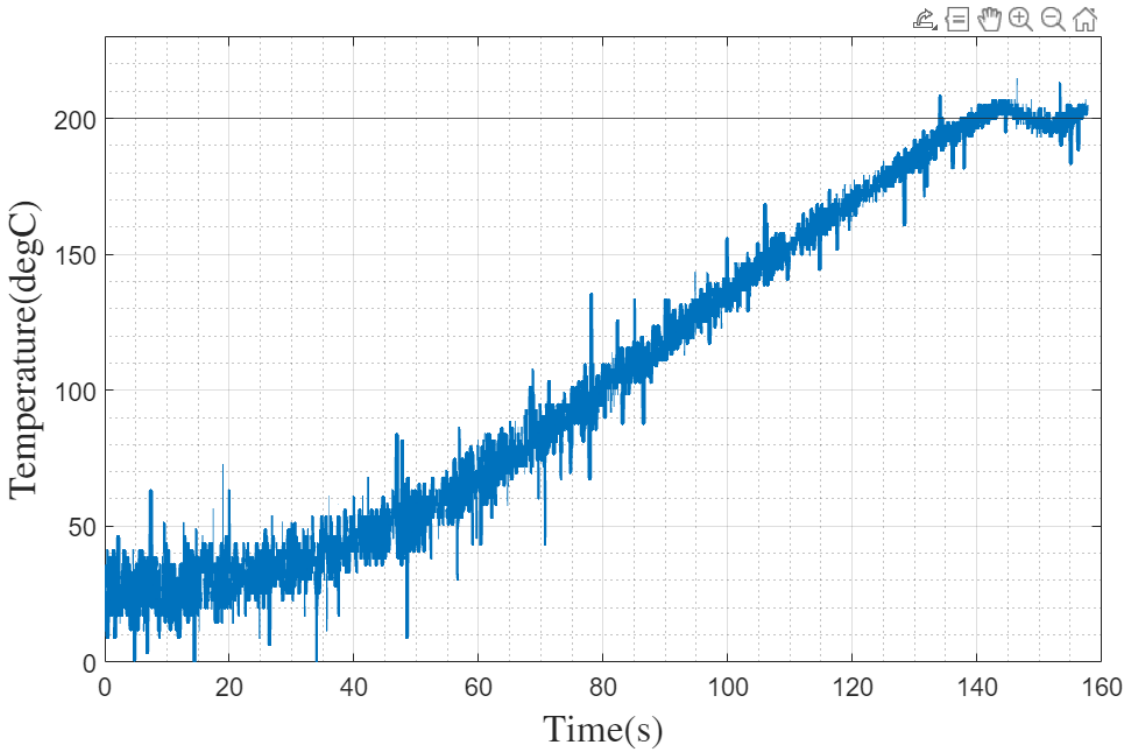


Figure 22 - Temperature profile for the hot end.

Figure 23 shows a denoised version of the plot. While there is still some noise present, particularly at the lower temperatures, the trend is clear enough that it can be interpreted.

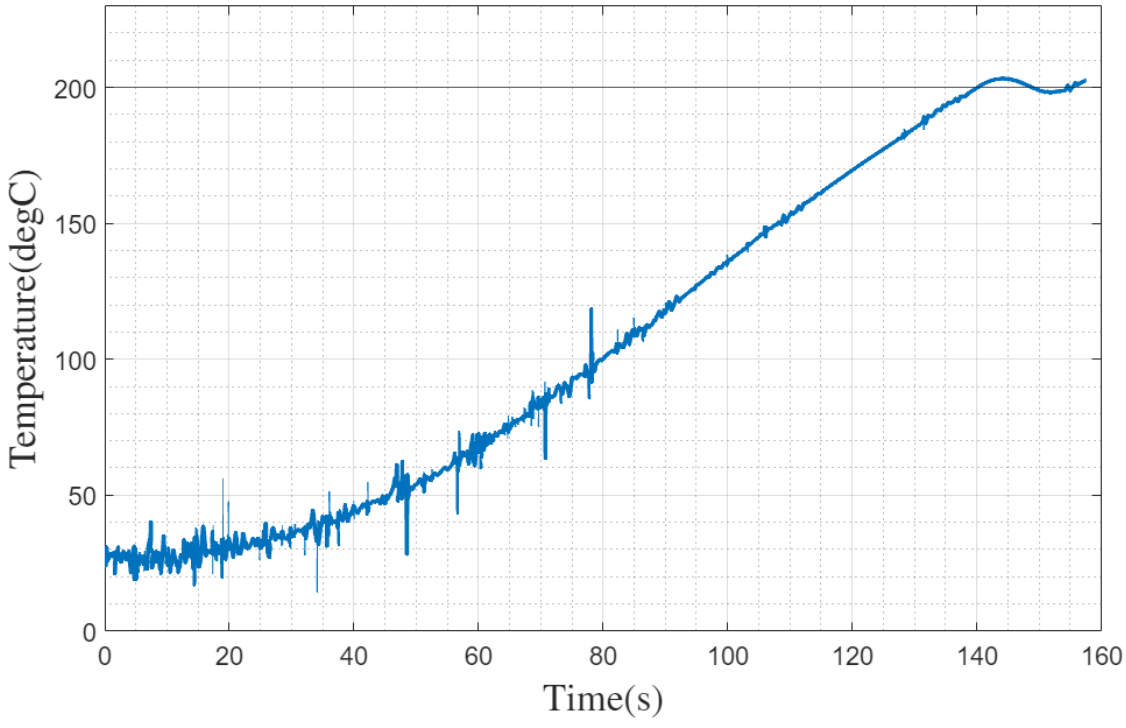


Figure 23 - Denoised temperature profile for the hot end.

Figure 24 shows a denoised plot of the hot end at steady state, after being heated for 5 minutes.

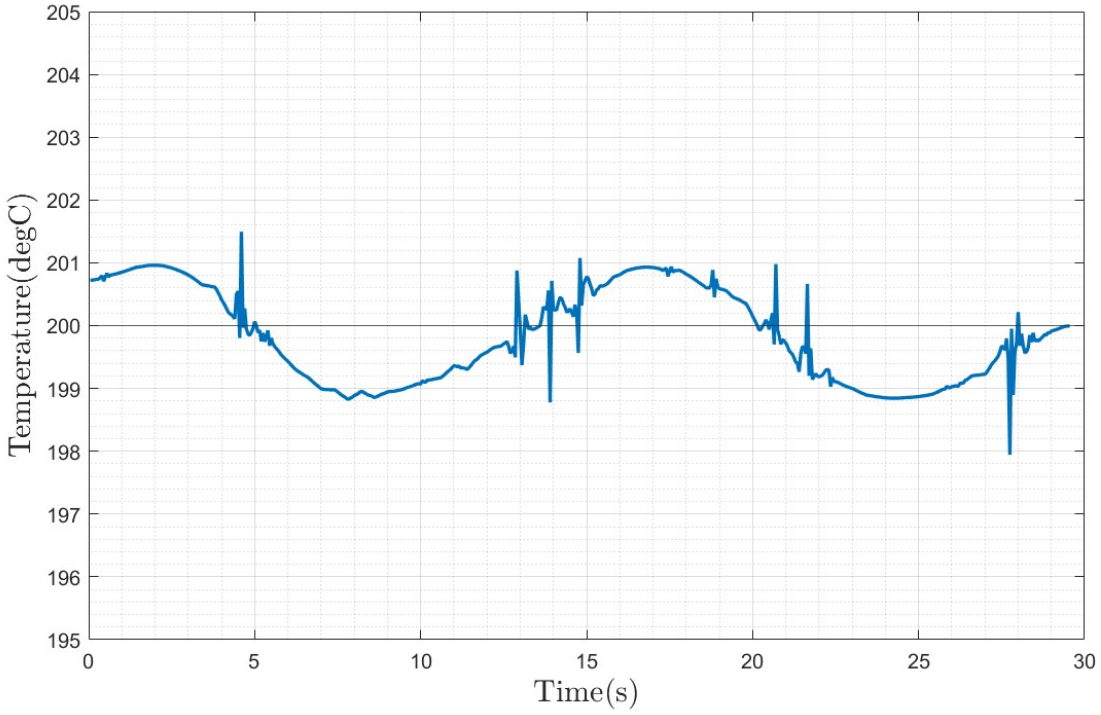


Figure 24 - Hot end temperature profile at steady state.

2.2.2. Speed Control

The stepper is controlled by a DM856 stepper driver, which takes input signals for direction and speed [4]. When the direction input is low, the motor feeds filament forward, and conversely when the input is high. The speed input is taken as a pulse rate, with a higher number of pulses per second relating to a higher feed speed. Both these inputs are provided from an Arduino, which oversees all print head control. The Arduino outputs a pulsed signal by implementing a delay between pulses. Delay is calculated as the inverse of pulse rate.

To find the relationship between pulse rate and feed speed, a series of tests were performed. Arbitrary pulse delays of 50ms, 40ms, 30ms, 20ms, 10ms, 5ms were initially tested as a baseline. A point on the filament was marked, and the distance between the mark and the point where the filament entered the extruder was measured. The extruder was then run for ten seconds with the target pulse delay using an Arduino program, which can be found in Appendix 3, and the distance was again measured. The difference between these two measurements was recorded and was repeated three times for each pulse delay, with an average value calculated. The feed speed can then be calculated by dividing length by time. Table 4 shows the results of the initial tests.

	Filament Extruded (mm)					
	Trial					
Delay (ms)	1	2	3	AVG	speed(mm/s)	Pulse rate(1/s)
50	4.1	4.1	3.95	4.05	0.405	20
40	5.37	5	5.04	5.13667	0.513666667	25
30	6.83	6.28	6.95	6.68667	0.668666667	33
20	10.13	10.06	10.07	10.0867	1.008666667	50
10	20.42	20.26	20.29	20.3233	2.032333333	100
5	40.79	40.53	40.53	40.6167	4.061666667	200

Table 4 - Uncalibrated print head test results

Speed can then be plotted against pulse rate with a linear relationship, as shown in Figure 25. The trend line can be characterised as shown in Equation 3. If it is assumed the trend line passes through the origin, the equation can be further simplified.

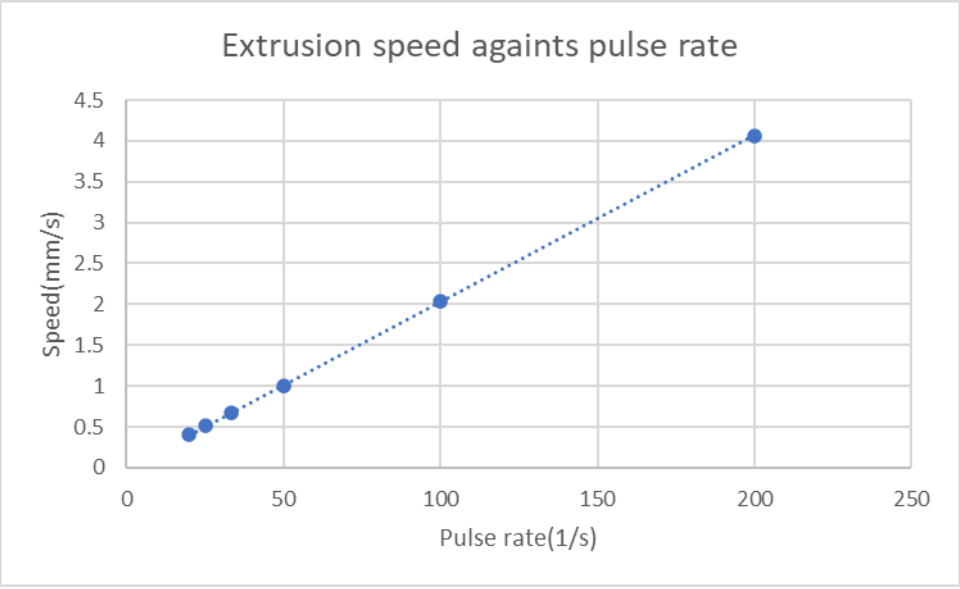


Figure 25 - Plot of Extrusion speed against pulse rate.

$$F = 0.0203 * P - 0.00111$$

$$F = 0.0203 * P$$

Equation 3 - Relationship between feed speed and required pulse rate.

3D printer commands are usually given in terms of the movement speed of the print head, and not the feed speed of the extruder motor. Since the nozzle diameter is smaller than the diameter of the filament, but the flow rate into the nozzle is equal to the flow rate out, it can be summarised that the print head movement speed is greater than the feed speed. If the cross section of a printed line is assumed to be square, with a width equal to the nozzles diameter and a height equal to the layer height, as shown in Figure 26, the flow rate out of the nozzle can be assumed to be the cross-sectional area multiplied by the print head speed.

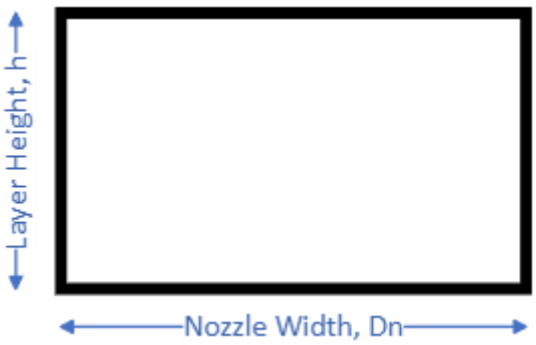


Figure 26 - Assumed cross section of extruded filament.

Hence, since flow rate in is equal to flow rate out, Equation 4 can be derived, where F is the feed speed in mm/s, A is the cross-sectional area of the filament in mm^2, U is the print head speed in mm/s, D_n is the nozzle diameter in mm and h is the layer height in mm.

$$F * A = U * D_n * h$$

Equation 4 - Equation relating flow into the nozzle to flow out.

This can be rearranged to find feed speed as a function of print head speed, as shown in Equation 5. For the system, the nozzle diameter is 0.4mm, layer height is 0.2mm and filament cross sectional area is $2.405mm^2$. Hence, the equation can be further simplified as shown.

$$F = \frac{U * D_n * h}{A}$$

$$F = U * 0.392$$

Equation 5 - Relation between print head speed and feed speed.

By substituting Equation 3 into Equation 5, Equation 6 can be generated, which expresses pulse rate as a function of print head speed. An arbitrary constant c has been included, which acts as a scaling factor to account for any discrepancy between theoretical and practical results. This is sometimes referred to as a flow multiplier.

$$P = U * 1.638 * c$$

Equation 6 - Relationship between pulse rate and print head speed.

Using this equation, the delay length for a series of print head speed could be calculated, as shown in Table 5. Print head speeds between 10mm/s and 70mm/s were chosen since they represent the operating window of a standard desktop printer.

Head speed (mm/s)	Flow (mm3/s)	Feed speed (mm/s)	Pulse rate (s^-1)	Delay (ms)
10	0.8	0.33	16.39	61.03
20	1.6	0.67	32.77	30.51
30	2.4	1.00	49.16	20.34
40	3.2	1.33	65.54	15.26
50	4	1.66	81.93	12.21
60	4.8	2.00	98.32	10.17
70	5.6	2.33	114.70	8.72

Table 5 - Calculation of required pulse delay from target print head speed.

The initial test could then be rerun using these delays. The expected length to be extruded was calculated by multiplying feed speed by the run time. The experimental values could then be compared to the expected value to assess the degree of accuracy in the print head speed control. Table 6 shows the results of the final tests.

		Actual length				
Head speed(mm/s)	Expected length (mm)	1	2	3	AVG	Difference (mm)
10	3.33	2.97	3.22	3.33	3.17	-0.153
20	6.65	6.59	6.92	6.59	6.70	0.047
30	9.98	9.96	9.83	9.59	9.79	-0.186
40	13.31	13.43	13.49	13.24	13.39	0.081
50	16.63	16.87	16.66	16.87	16.80	0.168
60	19.96	19.71	19.7	20.18	19.86	-0.095
70	23.28	23.51	23.16	23.1	23.26	-0.028

Table 6 - Comparison between expected extruded length and experimental values.

2.3. Rapid Code

Rapid is the programming language which is used to control the robot arm. Any programs need to be written up and downloaded directly onto the robot controller, where they can be run sequentially. 3D objects need to be broken down into a toolpath and converted into Rapid code. Communication also needs to occur between the print head and the robot controller to ensure movement and extrusion speed are synchronised. Code was tested first in simulation, before being loaded onto the machine.

2.3.1. Pathing

An STL file is broken down into a tool path using a standard 3D printing slicer. PrusaSlicer was used for this project, since it was the most familiar and easy to set up. The slicer outputs the toolpath in the form of GCode, which needs to be converted to Rapid before being uploaded to the robot arm. GCode states movement commands and coordinates in a single line, whereas Rapid predefines the set of coordinates that are used within a module, then executes the movements within the called procedure. As such, the conversion process needs to generate a set of targets and a set of movement commands individually.

An existing Python script was modified, which scrapes an inputted GCode file to look for movement commands, and outputs two individual text files containing the Rapid targets and movement commands [82]. The full code can be found in Appendix 4. Figure 27 shows how the script searches for relevant commands within the GCode file.

```
if len(temp) == 0:
    temp = [";"]

# if temp[0] ==";LAYER_CHANGE":

#     dz = round(dz + 0.2, 2)

if temp[0]=="G0" or temp[0]=="G1":

    x = 0.0
    y = 0.0
    z = 0.0
    f = 0.0
    e = 0.0
    dx = 0.0
    dy = 0.0
    df = 0.0

    for comp in temp:
        #   look for speed commands
        if comp.startswith("F"):
            df = round(float(comp[1:]) / 100)
            dx = positions.iloc[-1].x
            dy = positions.iloc[-1].y
        #  look for X coordinates
        if comp.startswith("X"):
            dx = float(comp[1:])
            dz = positions.iloc[-1].z
            df = Speed.iloc[-1].f
        #  look for Y coordinates
        if comp.startswith("Y"):
            dy = float(comp[1:])
            dz = positions.iloc[-1].z
            df = Speed.iloc[-1].f
        if comp.startswith("Z"):
            dx = positions.iloc[-1].x
            dy = positions.iloc[-1].y
            df = Speed.iloc[-1].f
            dz = float(comp[1:])
```

Figure 27 - Code snippet showing how code is scraped for movement commands.

The script searches lines of code which start with G0 or G1, identifying a movement. It then searches the following text for the characters X, Y or Z, and identifies any coordinates attached to them. These coordinates are then stored in an array. If a coordinate isn't specified, such as where a point occurs at the same Z value, the previous value is inserted into the array. These coordinates are then written up from the array into the output files, surrounded by the relevant syntax for Rapid. The content from these files can then be written into a module in Rapid editor using the following format.

```
MODULE "moduleName"
"contentFromTargetsOutput"
    PROC "procedureName"
    "contentFromMovementsOutput"
    ENDPROC
ENDMODULE
```

Figure 28 - Module format for inputting conversion files into Rapid.

The module for each printing procedure can then be called in the main module, a full example of which can be found in Appendix 5. The controller only has a specified set of speeds by default, and so to use a range, program speeds need to be defined within the main module. Figure 29 shows how the speeds have been defined, which range from 1mm/s to 150mm/s, covering the full range required for 3D printing.

```
VAR speeddata v0 := [ 0, 500, 5000, 1000 ];
VAR speeddata v1 := [ 1, 500, 5000, 1000 ];
VAR speeddata v2 := [ 2, 500, 5000, 1000 ];
VAR speeddata v3 := [ 3, 500, 5000, 1000 ];
VAR speeddata v4 := [ 4, 500, 5000, 1000 ];
VAR speeddata v5 := [ 5, 500, 5000, 1000 ];
VAR speeddata v6 := [ 6, 500, 5000, 1000 ];
VAR speeddata v7 := [ 7, 500, 5000, 1000 ];
VAR speeddata v8 := [ 8, 500, 5000, 1000 ];
VAR speeddata v9 := [ 9, 500, 5000, 1000 ];
VAR speeddata v10 := [ 10, 500, 5000, 1000 ];
```

Figure 29 - Declaration of speeds within Rapid module.

The heap size on the controller is limited to 8MB, which corresponds to around 15,000 coordinates loaded onto the controller. This causes an issue with larger models, which can contain over 100,000 coordinates. To deal with this, the larger hard storage of the controller can be utilised. By splitting the procedure for larger models across several modules, containing up to 7000 coordinates such that two modules can be loaded concurrently, each module can be individually loaded and unloaded onto the heap during the printing operation, as shown in Figure 30.

```
Load \Dynamic, "HOME:/BenchyMod1.MOD";
%"Benchy1"%;
UnLoad "HOME:/BenchyMod1.MOD";

Load \Dynamic, "HOME:/BenchyMod2.MOD";
%"Benchy2"%;
UnLoad "HOME:/BenchyMod2.MOD";

Load \Dynamic, "HOME:/BenchyMod3.MOD";
%"Benchy3"%;
UnLoad "HOME:/BenchyMod3.MOD";
```

Figure 30 - Loading and unloading procedure for large coordinate print programs.

2.3.2. Communication

The controller needs a way of communicating the movement speed of the arm back to the print head controller, such that their speeds can be synchronised. The outputs from the robot controller come from the built in DSQC 652 I/O board, which features 16 digital input ports and 16 digital output ports, shown in Figure 31 [83].

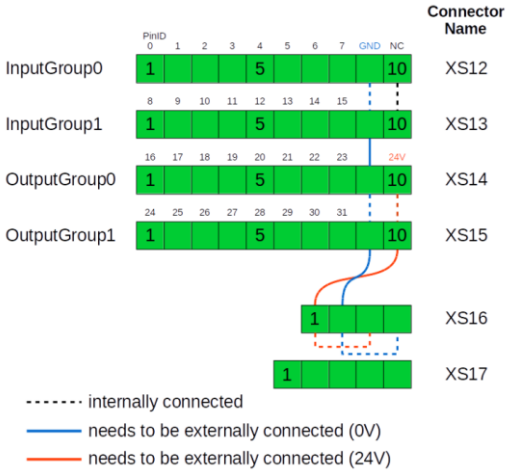


Figure 31 - Schematic of DSQC 652 digital I/O [83].

Since there are no analogue outputs, and generating a pulsed output may generate a delay in the execution of the code, it was chosen to output the speed in the form of a 7-bit binary number, utilising ports DO10 through to DO16 as each of the individual bits. DO9 was utilised as a retract signal.

To set the I/O to the correct value during operation, each movement instruction requires an additional output instruction to write speed and retract signals. Additional code was added to the python script to parse the target GCode for speed and retraction commands, and to save them to a relevant array, shown in Figure 32 and Figure 33.

```
#   look for speed commands
if comp.startswith("F"):
    df = round(float(comp[1:]) / 100)
    dx = positions.iloc[-1].x
    dy = positions.iloc[-1].y
```

Figure 32 - Code snippet to parse GCode for speed commands.

```
#   look for retraction commands
if comp.startswith("E-.8"):
    de = 1
    dx = positions.iloc[-1].x
    dy = positions.iloc[-1].y
#   look for detraction commands
if comp.startswith("E.8"):
    de = 0
    dx = positions.iloc[-1].x
    dy = positions.iloc[-1].y
```

Figure 33 - Code snippet to parse GCode for retraction and detraction commands.

These were then written in the output file before the respective movement command, such that as the movement is executed, the speeds of the arm and extruder are synchronised, shown in Figure 34.

```
# Writes the ABB move commands into a file
def writeMoveL(i, Speed, Retraction):
    f = int(Speed.f)
    r = round(Retraction.e)
    if r == 1:
        fb = 0;
    if r == 0:
        fb = f;
    fb = format(fb, '07b')

    f64 = fb[::-1][6]
    f32 = fb[::-1][5]
    f16 = fb[::-1][4]
    f8 = fb[::-1][3]
    f4 = fb[::-1][2]
    f2 = fb[::-1][1]
    f1 = fb[::-1][0]

    string1 = "MoveL "
    string2 = "d" + str(i)
    string3 = ",v" + str(f) +", fine, Hemera\WObj:=PrintBed;"
    string4 = "\n"
    string5 = "setDO D652_10_DO10, " + str(f64) + "; "
    string6 = "setDO D652_10_DO11, " + str(f32) + "; "
    string7 = "setDO D652_10_DO12, " + str(f16) + "; "
    string8 = "setDO D652_10_DO13, " + str(f8) + "; "
    string9 = "setDO D652_10_DO14, " + str(f4) + "; "
    string10 = "setDO D652_10_DO15, " + str(f2) + "; "
    string11 = "setDO D652_10_DO16, " + str(f1) + "; "
    string12 = "setDO D652_10_DO9, " + str(r) + "; "

    moveL =  string5 + string6 + string7 + string8 + string9 + string10 + string11 + string12 + string4 + string1 + string2 + string3 + string4
    moveLs.append(moveL)
```

Figure 34 - Code snippet for writing digital outputs before movement commands.

Since the DSQC 652 records a 24V signal, and the print head controller records 5V, the signal voltages need to be scaled for communication in either direction. For voltage scaling down from 24V to 5V it was chosen to use an optocoupler. This completely isolates the 24V rail from the 5V rail, which means there is no chance of damaging the microprocessor. Since this is not a risk in the other direction, it was chosen to use a relay to switch a 24V bus upon a 5V input. This was chosen since it was the cheapest and easiest to implement solution, and it has a relatively consistent and low switching time, preventing any signal delay. This was particularly important given that these signals were used for bed levelling.

To convert the robot controller outputs back into an integer number at the print head controller, an existing Arduino program was used, and can be found in Appendix 6. This assigns a bit value to each signal line and sums the value of each bit based on whether the line is high. It then returns the sum value, calculates the required pulse delay based on the previously established relationship, and sets the print head speed. Upon receiving the retract signal, the direction is reversed, and the motor signal is pulsed 29 times with a delay of 17.44ms, causing a retraction of 0.8mm over 500ms.

The binary input and speed control programs were combined to create an overarching print head control program for the Arduino, which can be found in Appendix 7. This runs alongside the Rapid path program and synchronizes the extrusion and movement speeds.

2.4. Print Testing

The performance of the system was assessed through a series of print tests. These were done to measure and calibrate various factors, including line width, Z offset, dimensional accuracy and print quality. This was done using a 2D (Two Dimensional) line and square test, and a 3D Calibration Cube and Benchy test.

2.4.1. Line Test

This test had the system print a 2D, continuous, zigzagging line across a 200mm X 200mm square in the centre of the print bed, at a speed of 10mm/s. This was used as an initial system check, but also allowed for measurement of line width consistency and Z offset. The results of the test can be seen in Figure 35. The toolpath for this print was generated manually in Rapid.

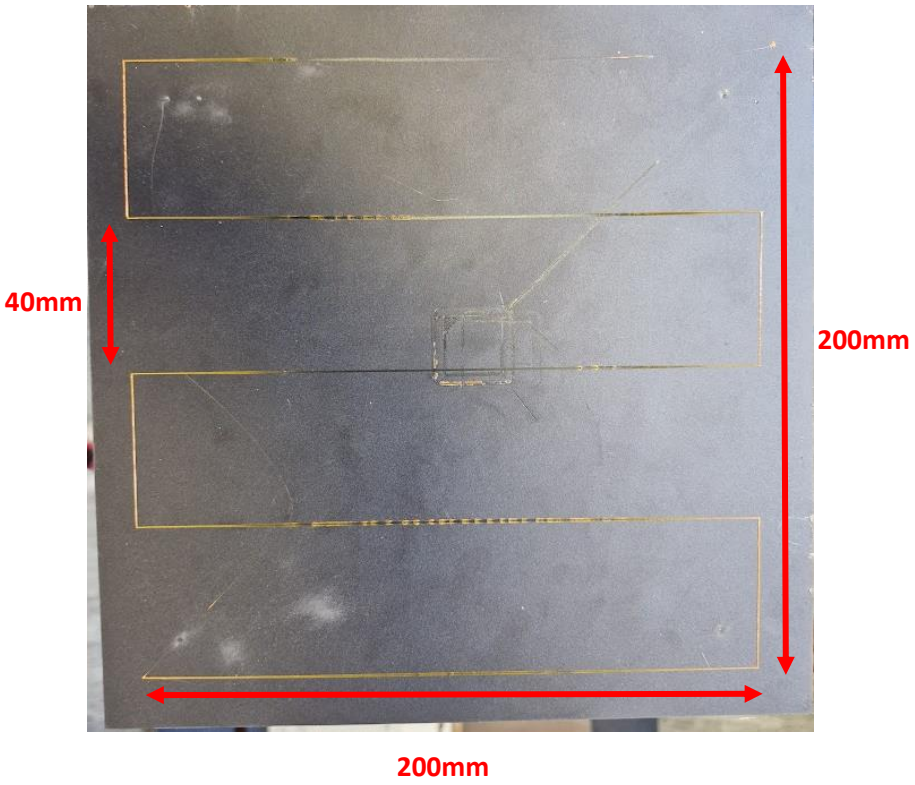


Figure 35 - Results of line test.

The printer could replicate the design pattern and reached coordinates with a reasonable degree of accuracy. Line width was measured to be 0.4mm at the edges, in the region where the print bed had been levelled. However, as the path crosses the middle of the bed, the line width can be seen to increase significantly to 0.8mm, indicating that the distance between the nozzle and the bed decreases in this region.

2.4.2. Square test

This involves printing a filled in, 2D square in the centre of the print bed, with dimensions of 20mm X 20mm. This was used to calibrate the Z offset required to achieve a consistent first layer. The print was also checked for dimensional accuracy. Figure 36 shows the results from testing, after iterating several times to find the required Z offset of +0.15mm. The toolpath for this test and subsequent tests was generated using the GCode to Rapid converter.

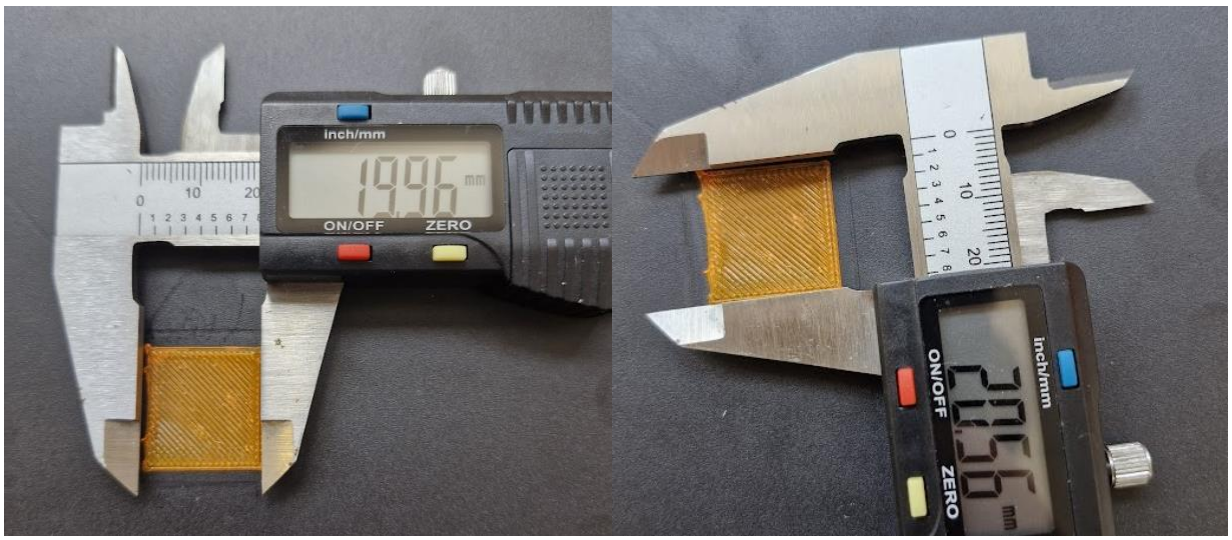


Figure 36 – Results from square test.

The results show a solid fill in the centre of the square, with consistent line width, resulting in no gaps or bulges across the bed. This indicates a correct Z offset at the centre of the bed. The print has dimensions of 19.96mm across the x direction, and 20.56mm across the y direction.

2.4.3. Calibration Cube Test

This was the first 3D test performed. The Calibration Cube model is commonly used in desktop printing to benchmark a printer's performance [84]. The model consists of a 20mm X 20mm X 20mm cube, with the characters X, Y and Z inlaid on the faces. The test proves the capability of printing 3D objects, as well as checking the dimensional accuracy over a complete print, and ability to handle fine detail. Figure 37 shows the first attempt at printing the model.

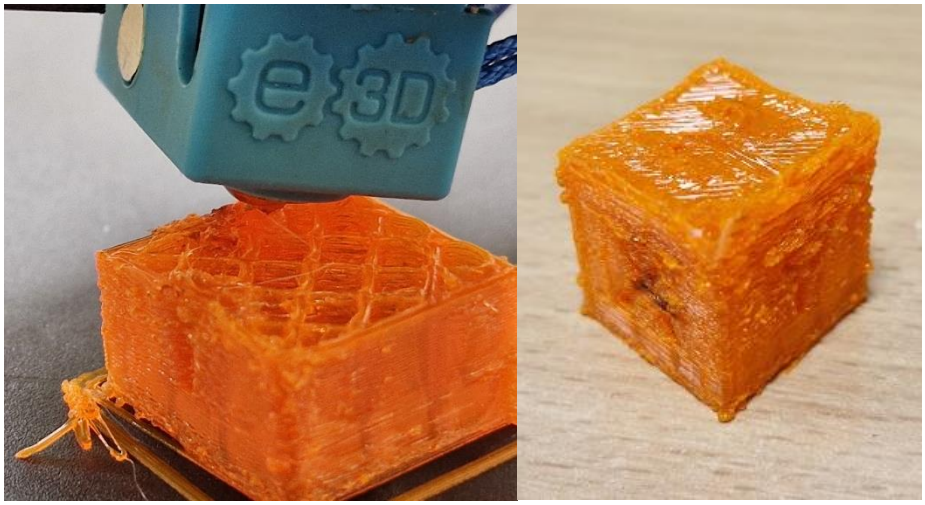


Figure 37 - Results from initial calibration cube test.

The system was able to print the entire model, however the result has a very poor surface finish, with most of the detail being lost. This was due to a lack of cooling, resulting in the movement of lower layers as new ones were added on top, distorting the print. To counteract this, cooling capability was added to the system, using a 5015-blower fan and a modified cooling duct [85]. Bed adhesion was also an issue, with several prints detaching themselves from the print bed before finishing, as shown in Figure 38.

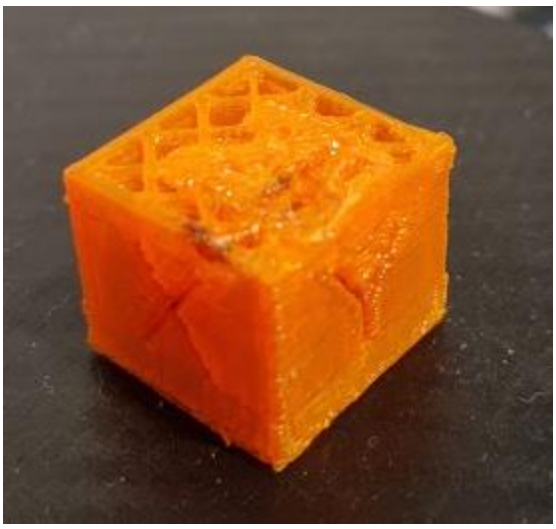


Figure 38 - calibration cube after detaching from print bed.

This was rectified by reducing the z offset to +0.12mm, slightly sacrificing the finish of the first layer for better adhesion. The final results can be seen in Figure 39.

Figure 39 - Final results from calibration cube test.

The print has a much better surface finish and can capture the inlaid surface detail. Measured dimensions in X, Y and Z are 20.23mm, 20.43mm and 19.59mm respectively.

2.4.4. Benchy Test

The final print test was the 3DBenchy [86]. This is a widely used test in the desktop 3D printing community, and has challenging geometry with multiple points of failure, which can help to better tune a printer's parameters. The print file contained over 10,000 individual coordinates, meaning the method of file splitting detailed in section 2.3.1. needed to be implemented. The results can be seen in Figure 40.

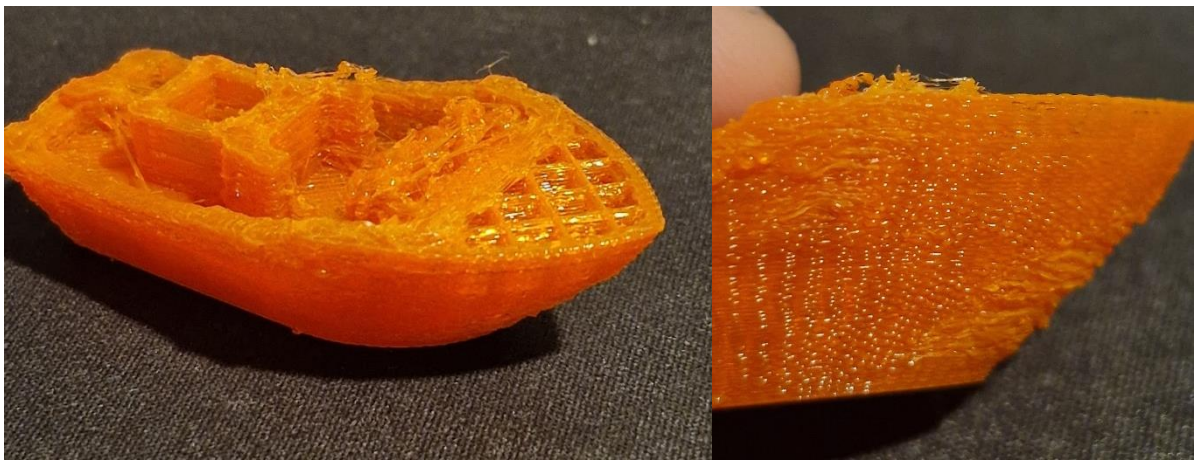


Figure 40 - Results of the Benchy print test.

Unfortunately, issues with the nozzle clogging prevented the production of a full Benchy. However, enough was produced to demonstrate successful file splitting. The robot arm was observed to have very jerky motion. This was due to how the robot arm processes fine zones; this causes the robot to come to a full stop at each coordinate, as shown in Figure 41.

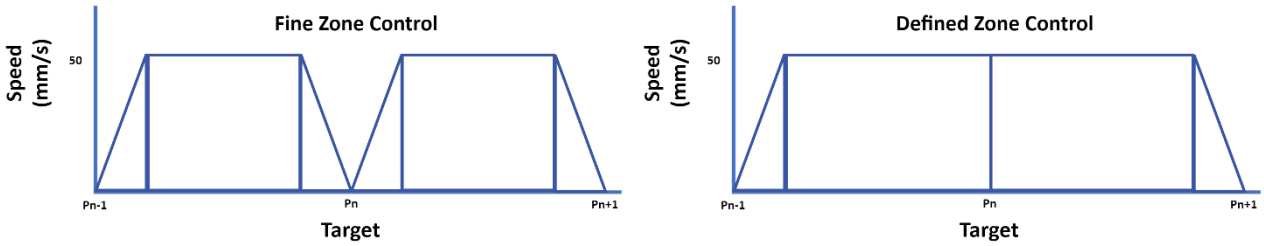


Figure 41 - Speed profile of end effector with fine control vs defined control zones.

Changing all zone control from fine to a defined value resulted in a higher quality completed print, as shown in Figure 42. A different reel of filament was also used since it resulted in better quality and bed adhesion.

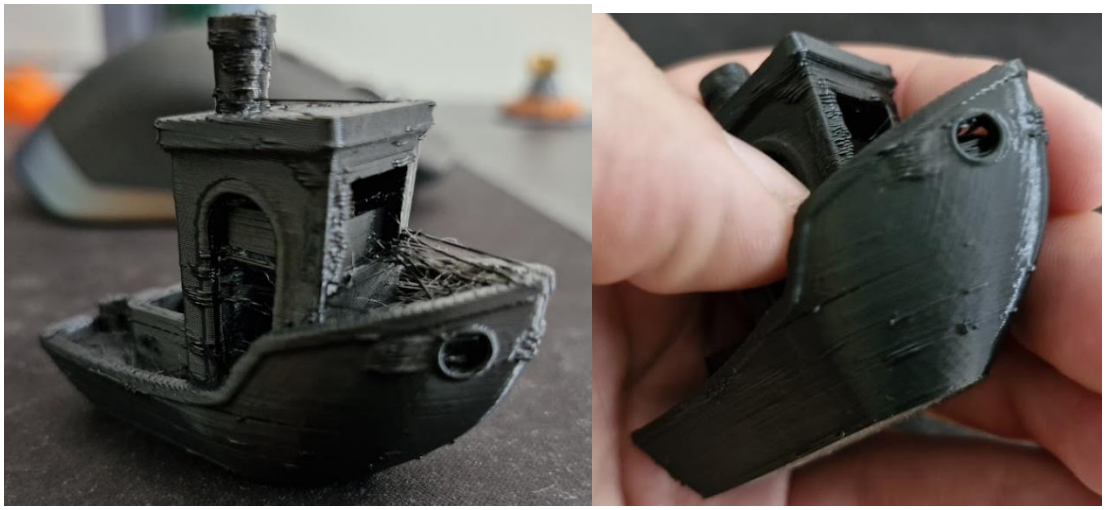


Figure 42 - Results from Benchy test after zone data and filament.

2.5. Bed Levelling

To account for any variances in heights that occur across the bed, the BLTouch sensor was used to perform autonomous bed levelling. This involves probing the bed at a series of points to generate a complete topology of the bed height. This is used to alter the Z height during a print to accommodate for height differences across the bed [38].

2.5.1. Bed Probing

For the system, it was chosen to map the bed using a 25-point grid, which covers the centre position, where the most printing will occur. 25 points captures a significant amount of detail without taking too much time to collect the data. Figure 43 shows how the grid has been divided into points, with the bottom left corner being point (0,0).

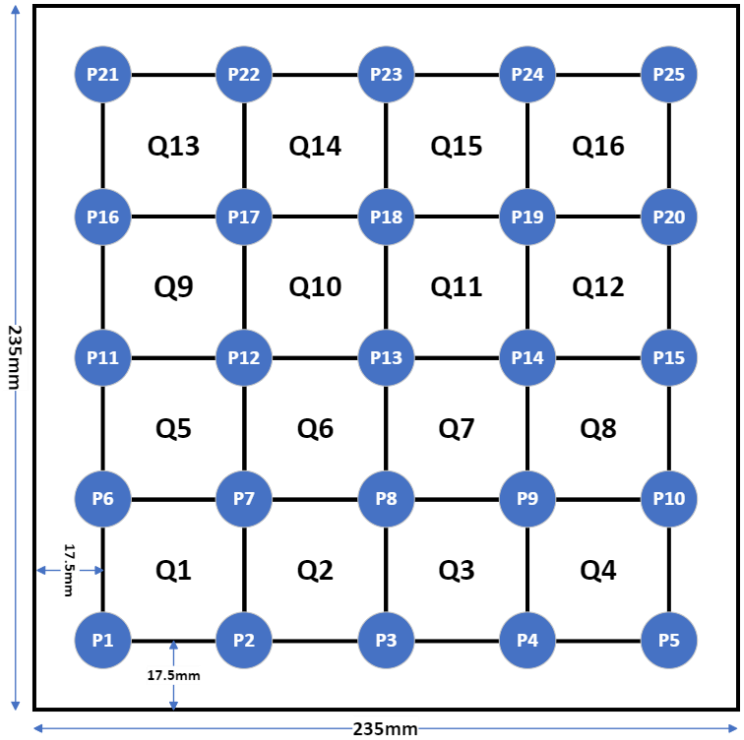


Figure 43 - Schematic of the probing pattern used for bed levelling.

Points P1–P25 represent the order in which the measurements are collected. Q1-Q16 represent quadrants within which individual points will be interpolated. The first point has been inset by 17.5mm such that each of the points are spaced 50mm apart.

The BLTouch bed level sensor takes a PWM (Pulse Width Modulation) control signal, which is used to set the state of the sensor, and provides a signal output, which indicates when a touch has occurred [5]. An existing Arduino program was edited that primes the sensor, records a touch and sets an output to high when it occurs, and resets the position of the probe [87]. This can be found in Appendix 8. The output is then sent to the relay, which when high, allows a 24V signal to pass to DI10, indicating a touch to the controller.

In conjunction with this, Rapid code was generated to probe the bed at each of the points, which can be found in Appendix 9. The SearchL command was used for this, which moves the end effector towards a given coordinate, and stops when a given input signal changes to high. The coordinate at which the signal occurred is stored in the designated output variable. Figure 44 gives an example of how probing was performed at a single point. This was repeated for each of the 25 points.

```
MoveL OFFS(c2,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp1, c2, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c2,0,0,20),v10, fine, Hemera\WObj:=PrintBed;
```

Figure 44 - Rapid code snippet for probing a single point.

Since coordinates for the system are relative to the position of the nozzle, the offset distance between the BLTouch sensor and the nozzle needed to be found. This was done by placing the extended probe tip outside the bounds of the bed, with the tip below the bed surface, and moving the arm towards the bed until a slight deflection was observed. This was repeated 3 times for the X and Y axis respectively, and the coordinate at which touch occurred has been recorded in Table 7.

	Offset measurement (mm)			
Axis	1	2	3	AVG
X	24.81	24.96	24.88	24.8833
Y	-38.93	-39.15	-39.08	-39.0533

Table 7 - Measurement of probe offsets in X and Y axis.

These offsets were then applied to the search coordinates, such that they were representative of the probe position instead of the nozzle position. Offset for the Z direction was taken to be 2.4mm which was based off adjusting the lowest recorded value to achieve a distance of 0.15mm between the nozzle and the bed.

A MATLAB script was used to plot the topology of the bed on a 3D graph from the data collected. The results are shown in Figure 45.

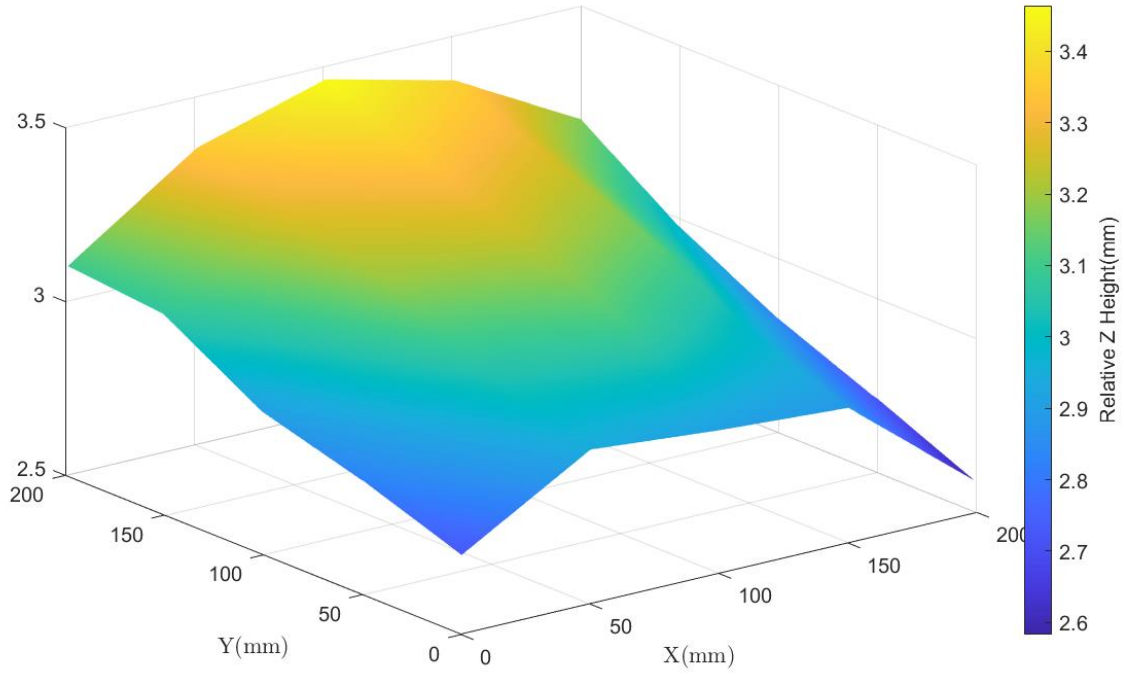


Figure 45 - Print bed topology from points recorded by BLTouch bed level sensor.

The bed shows a clear rise towards the centre and rear positions, in line with what was observed with the line test. The peak maximum and minimum values were recorded as 3.460mm and 2.5830mm respectively, with a peak-to-peak difference of 0.881mm

To verify these results, a map of the beds topology was taken using a high-resolution 3D scanner. This generates a point cloud of the bed's relative heights. A MATLAB script was used to plot these points, as shown in Figure 46.

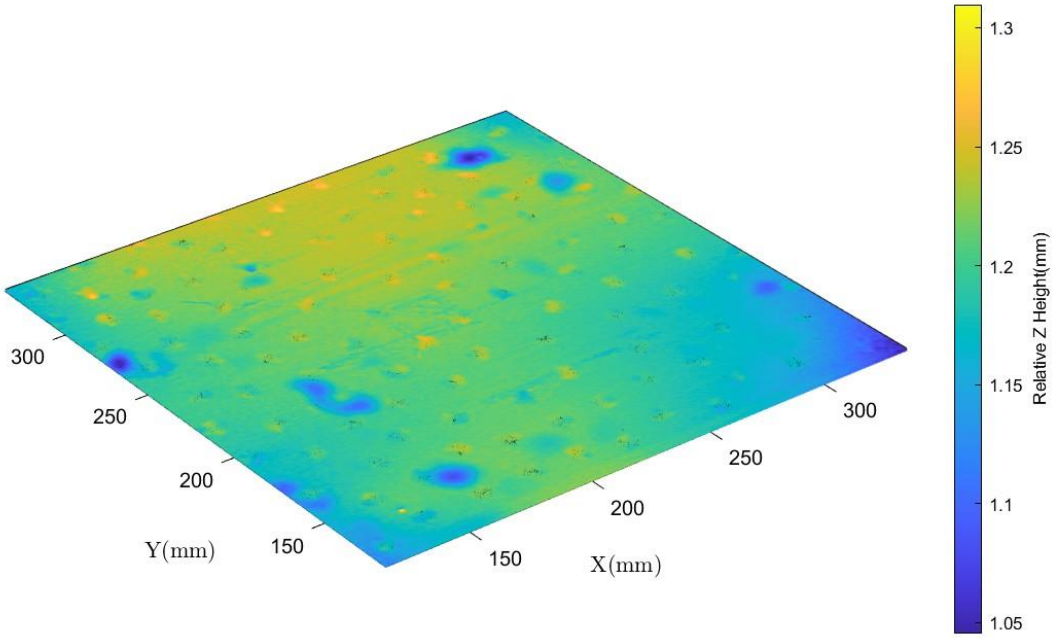


Figure 46 - Plot of bed topology from 3D scanner results.

The results show maximum and minimum values of 1.0454mm and 1.3097mm respectively, with a peak-to-peak value of 0.2643mm. The small yellow dots that can be seen on the plot represent the tracking dots that were used for location whilst performing the scan. Figure 47 shows how tracking dots were applied to the bed.

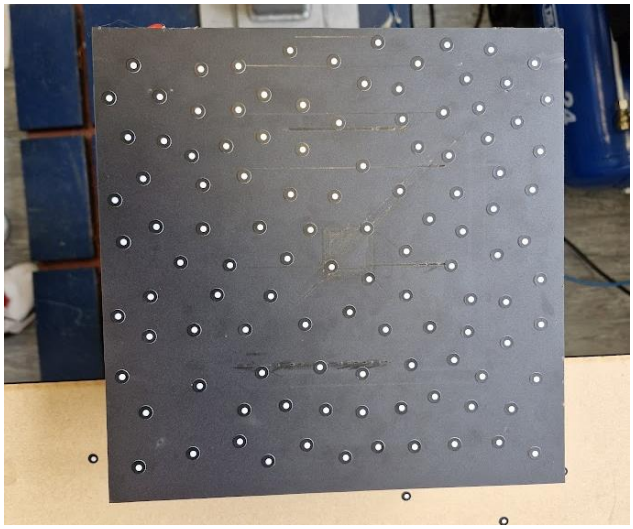


Figure 47 - Use of tracking dots for 3D scanning.

Due to the large discrepancy between results, the machine was completely recalibrated, and the probing procedure was repeated. Figure 48 shows the results after recalibration, with a maximum and minimum values of 3.1840mm and 3.0300mm, giving a peak-to-peak difference of 0.1540mm.

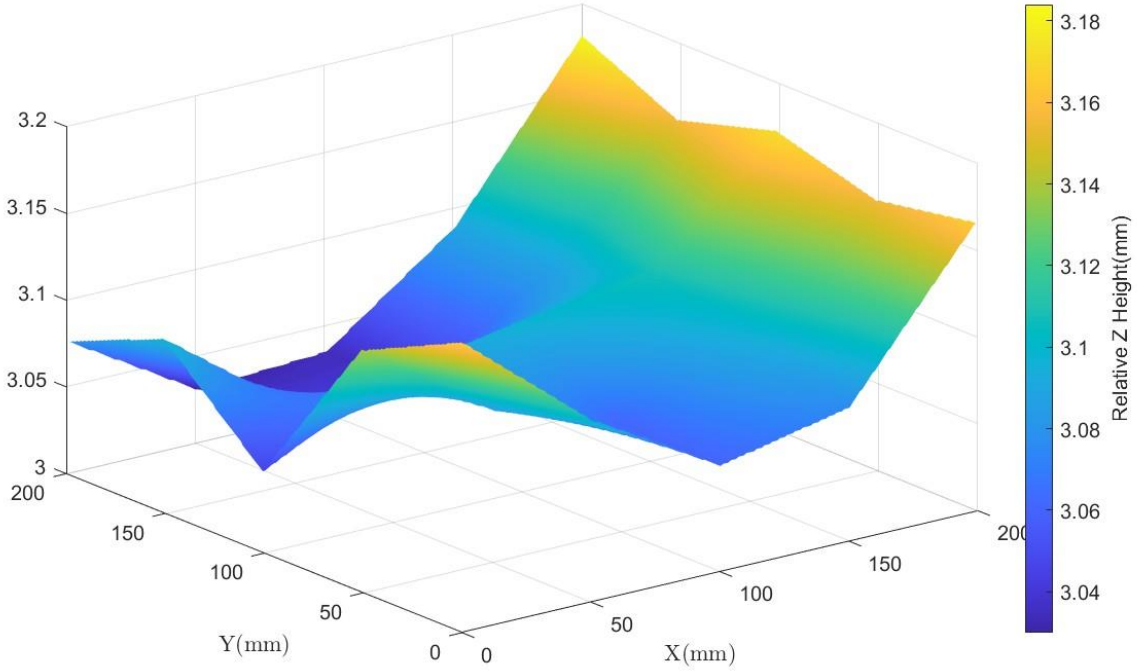


Figure 48 - Plot of bed topology after re-calibration.

2.5.2. Bilinear Interpolation

To perform bed levelling, a full map of the Z height at every coordinate on the bed need to be generated. This is done using bilinear interpolation. An existing python script was generated, which takes an inputted target coordinate, with a specified set of corner coordinates and Z heights and generates the height at that point. This script was incorporated into the GCode converter script, where it takes the 25 recorded bed points as input, and sweeps through each coordinate in a quadrant, for each quadrant, generating individual matrices of Z heights for each quadrant. This can be found in Appendix 10. These matrices were then combined to create an overall lookup table, providing the Z height at each integer coordinate.

These offsets were then added to the target Z height from the GCode during the output file writing process, with the previously measured 2.4mm difference between the probe point and nozzle subtracted. Since the bed level program only probed points within a 200mm X 200mm square in the centre of the bed, points outside this region had a standard 0.15mm offset applied.

Its desirable for the nozzle to be perpendicular to the plane of the print surface to achieve the best results and prevent collisions between the end effector and the print surface. As such, the angle of the print head needs to be adjusted for any surface variation. This is particularly important for more extreme bed angles. The relative orientation of the bed at a point was measured with reference to its neighbouring points, as shown in Figure 49.

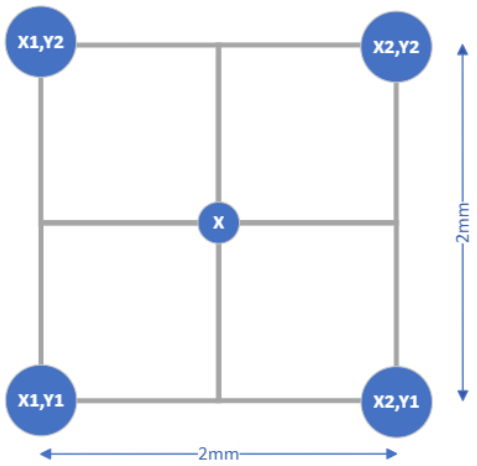


Figure 49 - Reference coordinates to find bed angles at point x.

Trigonometry was then used to calculate the relative angle of rotation of the bed along the X and Y axis, as show in Equation 7. Since two deltas are calculated, an average is performed as part of the angle calculation.

$$\Delta X_{lower} = (X_1, Y_1) - (X_2, Y_1)$$

$$\Delta X_{Upper} = (X_1, Y_2) - (X_2, Y_2)$$

$$\Delta Y_{lower} = (X_1, Y_2) - (X_1, Y_1)$$

$$\Delta Y_{lower} = (X_2, Y_2) - (X_2, Y_1)$$

$$\theta_X = tan^{-1}(\frac{\Delta X_{lower} + \Delta X_{Upper}}{4})$$

$$\theta_Y = tan^{-1}(\frac{\Delta Y_{lower} + \Delta Y_{Upper}}{4})$$

Equation 7 - Derivation of bed angles via trigonometry.

These angles can then be adjusted off the work object orientation of 180° over X and -90° over Z. Since Rapid code uses quaternion angles instead of Euler, the calculated angles were converted by implementing an existing python script. A code implementation of this process can be seen in Figure 50.

```
    dx1 = matrix[round(yr-18.5)][round(xr-18.5)] - matrix[round(yr-18.5)][round(xr-16.5)]
    dx2 = matrix[round(yr-16.5)][round(xr-18.5)] - matrix[round(yr-16.5)][round(xr-16.5)]
    dy1 = matrix[round(yr-16.5)][round(xr-18.5)] - matrix[round(yr-18.5)][round(xr-18.5)]
    dy2 = matrix[round(yr-16.5)][round(xr-16.5)] - matrix[round(yr-18.5)][round(xr-16.5)]

    Thetax = math.atan((dx1+dx2)/4)
    Thetay = math.atan((dy1+dy2)/4)

quart = euler_to_quaternion(-math.pi - Thetax, Thetay, -math.pi/2)
```

Figure 50 - Code implementation of angle calculation.

The line test was rerun to test the effectiveness of the bed levelling system, since bed height difficulties had been experienced previously. This test also covers the whole print bed, so shows that the levelling program is effective over the whole area. Figure 51 shows the result of the line test with bed levelling applied.

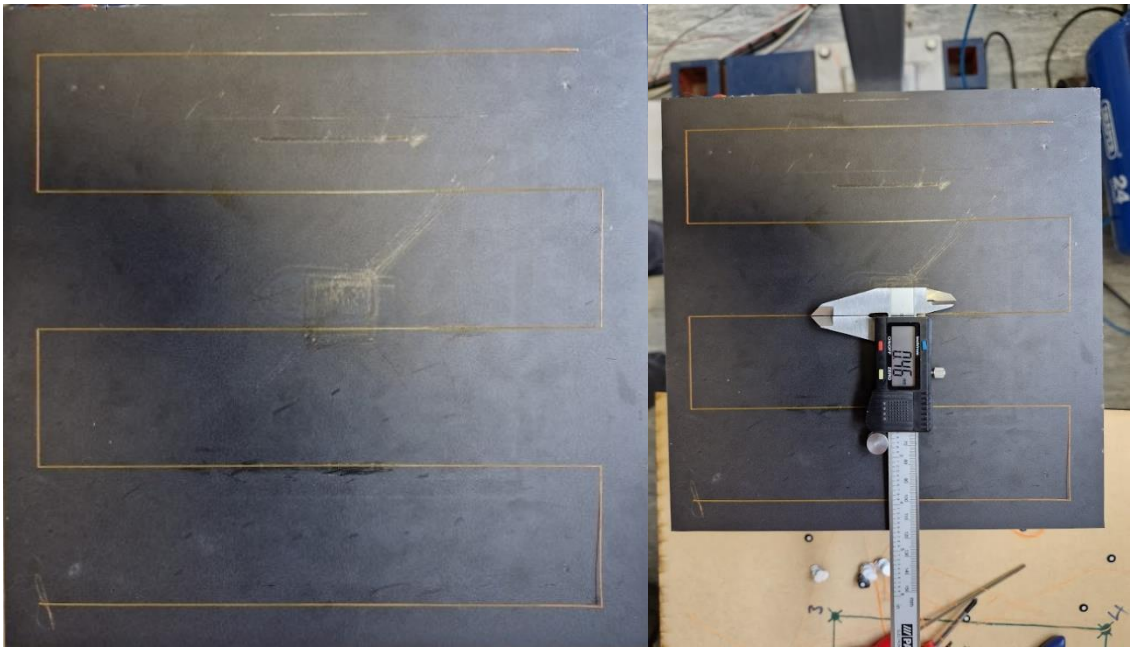


Figure 51 - Results from line test with bed levelling program applied.

As can be seen, a much more consistent line width is achieved throughout the print, measured at 0.46mm at all points. This shows that a constant distance between the nozzle and the print has been maintained, and the bed levelling program is effective.

3. Discussion and Conclusions

Table 8 shows a breakdown of the targets, with those met highlighted in green, and those not met highlighted in red. Only one target was not met, relating to the fitting of components within their respective safe zones.

Ref	Sub Ref	Tolerance	Target Met?	Tolerance Achieved
1	1.A	±1mm		Outside of safe zone by 50mm
	1.B	Joints maintain rotation of ±180°		Relevant joints rotate 180°+ in both directions
	1.C	Corner height adjusted ±5mm		±5mm adjustment achieved
2	2.A	±5°C		±1°C at steady state
	2.B	±0.5mm		±0.18mm
3	3.A	Line width 100% to 120% of the nozzle's diameter		0.46mm, 115% of nozzle width
	3.B	±0.5mm		±0.46mm
4	4.A	±0.06mm		±0.03mm
	4.B	±0.1mm		±0.6mm

Table 8 - Breakdown of the success of the project in meeting targets.

A full breakdown of the system's ability to reach each target has been provided, and conclusions have been drawn from these results. Advice has been given for potential next steps to improve system function or assess routes for further development.

3.1. Mechanical Integration

All components have been located within the safe zones specified in the user manual, however they exceed the bounds. Therefore, it cannot be said that target 1.A has been met. However, the robot retains full range of motion at the joints and can reach all points required for 3D printing in the given space. Therefore, target 1.B is fully met. Hence, the inability to meet target 1.A is irrelevant, since the robot can still operate fully without any interference. As such, the scope of target 1.A should be adjusted, such that it states that all components are to be mounted in a way that allows for suitable range of motion, without interference with mounted components, and allowing for a suitable margin of error. Use of a stock desktop printer bed provides the full range of adjustment required, meeting target 1.C.

The implementation of a wiring harness successfully keeps all connection safe from pinching and strain. However, with the cables bundled together, it can be hard to identify and isolate individual wires. This could severely impede a technician's ability to repair the system in the event of a component failure. Colour coding, labelling or heat shrinking of grouped wires could help with identification. An additional cable management system was also developed alongside this project by Aishwarya Bhattarai and Zaynab Cadow. An example of one of the prototypes can be seen in Figure 52.

Figure 52 - Example of a prototype cable comb.

This features slots for individual thinner wires, which mounts onto one of the larger robot control leads. This presents the wires neatly, and allows for quick and easy removal as required, whilst retaining the wires close enough to the machine.

The use of long, ungrounded wires presents the risk of EM interference. Whilst there have been no observed effects on the system during its operation, it is possible that EM interference may affect other nearby sensitive equipment. This presents a potential issue given there are multiple projects running in the same lab. If interference causes a significant disruption, additional testing should be conducted, and wires should be shielded. It's worth noting that this will severely impede the user's ability to manage and remove wires, so this step should only be taken once the system is fully established.

3.2. Print Head Control

The Hot end is able to reach and maintain its target temperature to ±1°C. This exceeds the tolerance stated in the target for ±5°C, meeting target 2.A. From the experimental results, extrusion lengths at all speeds within the operating window of a conventional desktop printer fall within ±0.5mm of the expected value. Therefore, it can be said that target 2.B is satisfied, and the print head controller has sufficient control over the print speed for accurate 3D printing. This was done with a 100% flow multiplier, showing that the derived equation holds exactly true for the testing conditions.

3.2.1. Temperature Control

Oscillating temperature is typical of bang-bang control, and implementing PID control instead would reduce variation at steady state. Implementing this method via a microcontroller, such as an Arduino, can also allow for adjustment of the target temperature, which will be required for printing with different materials. The control system could also be duplicated and implemented for a heated build plate, which will help with bed adhesion.

The temperature reading takes around 8s to cool back to the target temperature after an overshoot. This suggest there is a damping response present in the system, likely caused by the thermal mass of the hot end. Since the system inherently has quite heavy damping characteristics, achieving a constant temperature with PID control should be relatively straight forward. This can also help with printing at higher speeds, since higher flow rates will transfer heat out of the hot end quicker, and a delayed response will help prevent the temperature dropping below the target during an extrusion.

3.2.2. Speed Control

One possible cause for inconsistencies could be measurement error, due to the small length extruded. The nozzle temperature may also drop at the start of an extrude, and so a longer extrude would allow the temperature to settle to account for any initial drop. However, for longer extrudes it becomes hard to measure the length with the vernier callipers used, which have a maximum opening of 150mm. Hence, since the effects of this were negligible, results were deemed sufficient. If the system was to be used for print speeds greater than 70mm/s, further testing would be required.

For the experimental data, it was assumed that the flow into the nozzle is equal to the flow out, which is true when constantly extruding. However, when the printer performs a retraction, it pulls the filament up into the hot end, and molten filament can creep up slightly into the heat break. This can result in an apparent under extrusion, which will present itself as gaps or variance in line width within the print. There are other factors which may cause a lower extrusion rate than expected, including variance in the diameter of the filament, absorption of water into the filament, and variance in hot end temperature with changing speed and part cooling. To account for this, it may be necessary to increase the flow multiplier when performing print testing; desktop printers typically use a scaling factor between 100% to 120%. These factors cannot be calculated, and so the final adjustments to flow rate are tuned empirically when performing print testing.

3.3. Print Testing

The initial line test shows the line width varying from 0.4mm at the edge of the print bed, to 0.8mm at the centre. By accounting for inconsistencies in the systems kinematics using the bed levelling program, the line test was successfully repeated, showing a consistent 0.46mm line across the entire bed area. This represents 115% of the nozzle diameter, indicating a correct Z offset and a sufficiently consistent line width, meeting target 3.A.

The dimensions of the calibration cube were all within ±0.5mm of their target, with the largest variation being +0.43mm in the Y direction. The square test did show a variation in the Y axis of 0.56mm, which is slightly above the stated threshold. However, this is likely due to an increase in dimensions from first layer effects, such as elephants' foot, and the measurements over an entire 3D object will be more representative of the true dimensional accuracy of the system. As such, target 3.B has been met.

3.3.4. Print Quality

There are some small artifacts on the completed print that are indicators of poor performance. The calibration cube shows a clear bulging at the corners. This is indicating that the system is not retracting enough before coming to the end of an extrusion, and further tuning of the retraction distance would need to be done to reduce this effect. Prints also show warping at the first layer. This is due to the fan cooling the initial layer while molten material is being deposited on top, causing localised shrinking and curling. The best solution to this would be to use a heated bed, which will prevent the initial layers from cooling and help with bed adhesion.

It was noticed during the printing process for the Benchy that the filament spool was under a lot of tension, potentially limiting the extruders' ability to pass filament through to the nozzle. This is one potential cause for the halfway failure of the print. It was also noticed that the body of the extruder was particularly warm during the print, which can lead to heat creep, a potential cause of blockages. Further testing should be conducted with longer prints to observe if these issues persist.

The Benchy print also displayed webbing between parts of the print. This is due to inadequate retraction of filament before a movement. Further tuning of retraction length should help improve print quality, however this excess material can easily be removed, and is not a significant print defect.

Changing filament was seen to have significant impacts on bed adhesion and printability, with multiple prints failing using the orange filament. It is possible that the orange filament was an older reel, which had absorbed water over time. This causes the release of steam when printing, which reduces quality. It's also possible that the dyes used in the orange filament result in slightly different thermal properties, making printing more inconsistent.

Despite these minor issues, it can still be concluded that the system is capable of recreating models to a reasonable degree of accuracy and replicating the performance of a desktop printer. As such, the system is ready for further development for multi axis printing, however, may require additional cooling capabilities to be able to handle larger overhangs.

3.4. Bed Levelling

The robot controller provides Z height coordinates to 7 significant figures, implying a high degree of accuracy in the measurement. However, the initial results from the probing highly disagreed with that of the 3D scan produced, with peak-to-peak measurements disagreeing by 0.617mm. This was greatly improved by recalibrating the machine, which reduced the disagreement to 0.1103mm. However, it is likely that the peaks recorded in the scan data include the height of the stickers, which was measured to be around 0.14mm. This would reduce the disagreement further to 0.0297mm, well within the 0.06mm target, therefore meeting target 4.A. However, since the system was able to account for the initial discrepancy and complete the revised line test, it can be said that the system was able to account for an inaccuracy of 0.6mm, exceeding target 4.B by 0.5mm.

Figure 53 - Measured height of a tracking dot.

3.4.1. Bed Probing

The Z height at the end effector is calculated by forward kinematics, using joint angles recorded by rotary encoders. It is possible that an initial miscalibration of the machine from the factory, or a drift in the encoder values over time, caused the initial disagreement between the bed probing and the scan results. This was further confirmed with parameterisation testing performed by Ian Pearse, where the robot arm was found to miss a target position by over 1mm. This test involved having the robot repeatedly reach a series of target points, while its position was tracked by a laser scanner. The results can be seen in Figure 54, with the circles representing target positions, and the blue lines representing the travelled path of the robot.

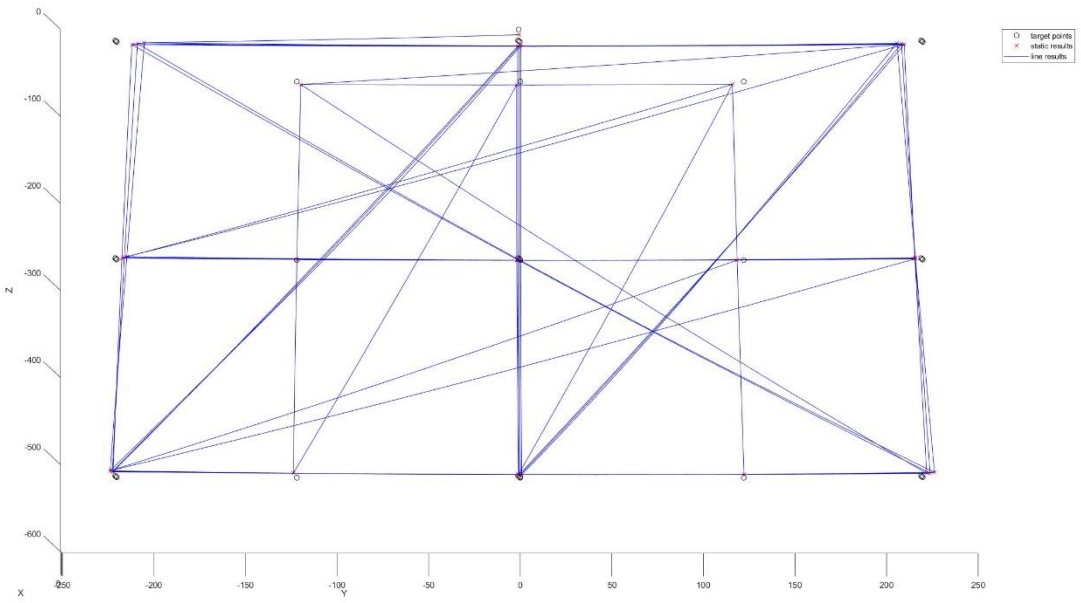


Figure 54 - Plot showing difference between true robot position and target robot position.

It can be seen that the inaccuracy gets much greater as the target gets higher. It can also be seen that the true position is lower than the target towards the centre of the bed, which would correspond with the apparent raise in bed heigh.

After calibration, the peak-to-peak heights agreed much more closely with that of the scan. Probed bed data appears flipped to the 3D scan data, which is due to how the measurements are taken. However, with this accounted for, the plots highly agree with each other. There was also an observed increase in first layer and overall print quality after calibration, suggesting that the overall lack of positional accuracy was affecting all aspects of the machine's performance. Additional tests to better parameterise the machine and to test the machines true positional accuracy after calibration would be recommended. Additional methods of improving accuracy may also need to be considered, including external guided motion.

3.4.2. Bilinear Interpolation

The script developed was able to take a series of probed measurements and adjust the generated GCode to account for variance, as well as adjusting the end effector angle to maintain a perpendicular orientation between the print head and the bed. This resulted in a significantly improved line test, with a constant line width of 115% of the nozzles diameter, showing the system's ability to maintain a constant distance between the bed and the nozzle. This test was performed while the machine was still in its uncalibrated state, showing the program could deal with a significant degree of bed variance, whether it's a result of the machines parameters or a true variance in bed height.

Given that the kinematics of the machine can result in high positional inaccuracies, this process should be considered a mandatory step in printing with the current system setup, particularly for prints which utilise a large bed area. With this process in place, the system should also be capable of further development for non-planar printing, involving printing on beds with a much more extreme angle, or a curved surface. Further testing should be conducted on the angle adjustment to assess the machines' ability to maintain an end effector orientation, since it may vary in a similar fashion to Z height.

4. Project Review

This section assesses the overall performance of the project was. Areas of the project that were particularly successful have been identified and discussed. Any potential setbacks have been summarised, with a description of their overall impact to the project, and how they were dealt with. Recommendations for future research and development of the system have been outlined based on conclusions made during development of the current system.

4.1. What Went Well

The system was able to successfully meet all of the target specified to replicate the performance of a desktop printer. As such, the system can be considered successful for implementing a fully functional robotic 3D printing system by retrofitting with off the shelf components.

The expenditure of the project is outlined in Appendix 11, with the order shown representative of the order in which each item was purchased. The total amount spent on the project comes to £218.10, £31.90 less than the £250 allocated. Spending was coordinated with the consideration that both me and Ian Pearse were assigned to the project with our own individual budgets. As such, the majority of the shared purchases were taken from my individual budget. This meant that Ian had significant headroom left on his individual budget, such that a large float was still available if any large purchases needed to be made.

Appendix 12 shows a Gantt chart, which is reflective of the actual project progress. The original Gantt chart can be found in Appendix 1. The completion of tasks was able to be parallelised much more than expected. This allowed for quicker progression at the start of the project, which provided additional time for tuning and testing, and was useful for data collection.

Component deliveries were organised to overlap with additional tasks wherever possible to avoid wasted time waiting. For example, when waiting for the initial components to begin integration onto the machine, work was already being completed for calibrating the print head and pathing the machine.

The project also had support from a group of students. They helped complete additional tasks, which allowed us to establish the machine in an accelerated timeframe. Tasks were published via a shared team's channel, where a detailed brief was provided that outline the task, the skills required, any measurements or considerations that would need to be taken, and who to contact. Figure 55 shows an example of a task brief. Meetings were also held with students, both online and in person, to get them up to speed with the task, and to provide any additional support and guidance where necessary.

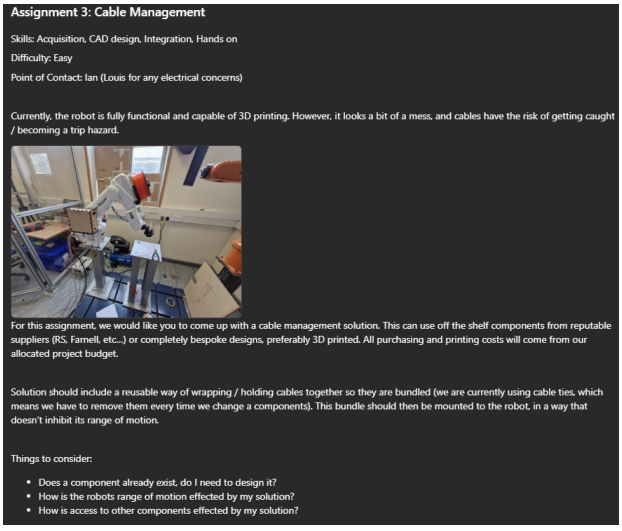


Figure 55 - Task brief posted in Teams channel.

4.2. Setbacks

There were several setbacks during the print testing and tuning process, resulting in failed prints. Poor bed adhesion resulted in multiple prints becoming unstuck from the print bed halfway through the printing process. This was somewhat rectified by running the initial layer height closer to the bed than recommended, resulting in much better adhesion, but sacrificing initial layer quality. There were also issues with the nozzle clogging, thought to be due to either binding of the filament reel causing a poor feed, or heat creep causing molten filament to rise up and solidify within the extruder. After removing and greasing the filament reel, no further issues were found.

It was initially assumed that part cooling would not be required for the scope of printing that was to be demonstrated with the system. However, after the first 3D test, it was apparent that print quality was suffering from a lack of cooling. This meant that an additional fan had to be purchased, resulting in a delay to print testing. A suitable cooling duct also had to be selected and modified, which took up time.

A few connections between components worked differently to expected, which resulted in additional redesign of the system. The print head controller that had been provided by the technician team provided insufficient power to the stepper driver, resulting in the unit failing when trying to heat the extruder and drive the stepper motor at the same time. This resulted in around a day's testing time lost to troubleshoot the source of the issue, initially believed to be an issue with the program running the stepper driver. This problem was rectified by using an external PSU to provide the required 30V to the driver.

The I/O from the robot controller also outputted a different voltage to what was expected. An initial assumption was made that it would output a 5V signal, as is standard with microcontrollers. However, upon review of the manual, and by verification with a multimeter it was found to send and receive 24V signals. This meant that additional time was required to search for and order additional components to scale the voltage for communication between components.

There was a two-week period where Ian required continuous access to the system for parametrisation and testing. During this time, I worked on processing and write up of results out of lab. Ian also had priority of the machine during the first few weeks, since this was required for mechanical integration. This allowed me to work independently outside of the lab on programming and simulation of the system, and calibration of the print head inside the lab, which was performed off machine.

4.3. Recommended Improvements and Testing

Implementation of a heated bed would significantly improve first layer quality and bed adhesion, which may also result in improved print quality. It would also allow for experimentation of different filaments other than PLA, which may be beneficial to proposed use cases for the machine. It would also be worth upgrading the print head controller to allow for digital PID control, partly to allow for the changing of the target temperature, and partly for reduced temperature variation.

Further improvements to print quality should also be investigated. Retraction settings should be better tuned such that the printer isn't trailing filament webbing between high points of a print. Additional more advanced printing parameters should be investigated, such as coasting with retraction, and linear pressure advance.

Significant accuracy issues between target robot positions and measured positions were found during bed levelling testing. This was found to be improved with manual calibration; however, it would be recommended to again perform a full positional sweep of the machine to assess any changes. If there is still a significant discrepancy, this factor should be accounted for during the conversion process of GCode to robot targets, such that the position the robot is reaching is representative of the true target position. External guided motion should be considered as an option to guarantee movement accuracy.

The bed level algorithm was demonstrated to account for a significant degree of height variance. As such further testing should be conducted to assess how extreme an angle the system is capable of measuring and accounting for. This will be a perfect first step towards nonplanar 3D printing.

Currently, files for larger prints have to be split manually into separate modules. Automating this as part of the main conversion code would help to simplify the process and save time.

5. References

[1] I. Gibson, D. Rosen, B. Stucker and M. Khorasani, “Development of Additive Manufacturing Technology,” in *Additive Manufacturing Technologies*, Cham, Springer, 2020, pp. 23-51.

[2] ABB, “Product specification IRB120,” 29 May 2019. [Online]. Available: https://library.e.abb.com/public/6aed5e91083f4fceb358eea2fe4c1bab/3HAC035960%20PS%20IRB%20120-en.pdf?x-sign=edIex5StIjpgmJBJJ95tak9NHdyuuut6mzzJHESGKt5i1JG8dhLRBdvggKptgBJn. [Accessed 25 March 2024].

[3] A. Reiswig, “E3D Hermes / Hemera (V6&Volcano),” GRABCAD, 06 December 2019. [Online]. Available: https://grabcad.com/library/e3d-hermes-hemera-v6-volcano-1. [Accessed 08 February 2024].

[4] Leadshine Technology Company Limited, “User’s Manual For DM856 Fully Digital Stepping Driver,” 2009. [Online]. Available: https://mecheltron.com/sites/default/files/webresources/ElectronicComponents/StepperMotorDrivers/Single_Axis_Drivers/pdf/DM856m_Mecheltron.pdf. [Accessed 18 March 2024].

[5] ANTCLABS, “BLTouch: Auto Bed Leveling Sensor for 3D Printers,” 05 April 2019. [Online]. Available: https://www.antclabs.com/_files/ugd/f5a1c8_d40d077cf5c24918bd25b6524f649f11.pdf. [Accessed 15 February 2024].

[6] L. Huygens, “Robotic 3D Printing for Complex Geometries: MEng Project Scoping & Planning Report,” University of Bath, Bath, 2024.

[7] S. Jin, W. Lian, C. Wang, M. Tomizuka and S. Schaal, “Robotic Cable Routing with Spatial Representation,” *IEEE Robotics and Automation Letters,* vol. 7, no. 2, 2022.

[8] megafonextra, “How to Level Your 3D Printer Bed Nozzle Height Calibration for the Best Printing,” Endurance Lasers, 23 April 2022. [Online]. Available: https://endurancelasers.com/how-to-level-your-3d-printer-bed-nozzle-height-calibration-for-the-best-printing/#:~:text=Use%20a%20feeler%20gauge%20to,0.2%20mm%20and%200.3%20mm.. [Accessed 06 February 2024].

[9] Prusa Research, “PID tuning,” 2022. [Online]. Available: https://help.prusa3d.com/article/pid-tuning_2265. [Accessed 2024 February 2024].

[10] H. Lin, K. Z. Ye and Y. H. Linn, “Modeling and Controlling of Temperature in 3D Printer (FDM),” in *IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Moscow, 2018.

[11] C. Simons and E. Grames, “Extruder Calibration: How to Calibrate E-Steps,” ALL3DP, 30 March 2023. [Online]. Available: https://all3dp.com/2/extruder-calibration-calibrate-e-steps/. [Accessed 02 February 2024].

[12] A. Sameer, L. Ahmad, M. Qawareeq and M. Salah, “Low-Cost 3D Printing Machine: Design, Implementation, Calibration, and Testing,” in *IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, Jordan, 2021.

[13] 3Faktur, “Accuracies and tolerances in 3D printing,” March 2023. [Online]. Available: https://3faktur.com/en/accuracies-and-tolerances-in-3d-printing/. [Accessed 07 February 2024].

[14] I. Grgi´c, M. Karakaši´c, H. Glavas and P. Konjatic, “Accuracy of FDM PLA Polymer 3D Printing Technology Based on Tolerance Fields,” *Processes,* vol. 11, no. 10, pp. 2810-2828, 2023.

[15] B. Tipary and G. Erdős, “Tolerance analysis for robotic pick-and-place operations,” *The International Journal of Advanced Manufacturing Technology,* vol. 117, no. 5, p. 1405–1426, 2011.

[16] Deckingman, “PRINTER BEDS – CONSTRUCTION, FLATNESS, LEVELLING AND COMPENSATION (IS IT NEEDED?),” HOT FILAMENT AND COLD COFFEE, 21 January 2017. [Online]. Available: https://somei3deas.wordpress.com/2017/01/21/printer-beds-construction-flatness-levelling-and-compensation-is-it-needed/. [Accessed 21 February 2024].

[17] R. B. Kristiawan, F. Imaduddin, D. Ariawan, Ubaidillah and Z. Arifin, “A review on the fused deposition modeling (FDM) 3D printing: Filament processing, materials, and printing parameters,” *Open Engineering,* vol. 11, no. 1, pp. 639-649, 2021.

[18] “PLA composites: From production to properties,” *Advanced Drug Delivery Reviews,* vol. 107, no. 3, pp. 17-46, 2016.

[19] E. Kopets, A. Karimov, L. Scalera and D. Butusov, “Estimating Natural Frequencies of Cartesian 3D Printer Based,” *Applied Sciences,* vol. 12, no. 9, pp. 4514-4542, 2022.

[20] Creality, “Ender-3 3D printer,” 10 April 2022. [Online]. Available: https://www.creality.com/products/ender-3-3d-printer. [Accessed 22 March 2024].

[21] Emvio Engineering, “BigRep ONE – Large scale 3D Printer,” 5 June 2022. [Online]. Available: https://emvioeng.com/shop/3d-printing/bigrep-one-large-scale-3d-printer/. [Accessed 22 March 2024].

[22] Bambu Lab, “Bambu Lab P1S 3D Printer,” 13 July 2023. [Online]. Available: https://uk.store.bambulab.com/products/p1s. [Accessed 27 March 2024].

[23] D. A. Maravi, G. M. Iparraguirre and S. R. Prado, “Implementation of a Digital PID Control for the Compensation of Loss Steps from CORE XY 3D Printer Motors Working at High Speeds,” in *IEEE ANDESCON*, Trujillo, 2020.

[24] J. O'Connell, “3D Printer PID Tuning: Simply Explained,” ALL3DP, 20 March 2022. [Online]. Available: https://all3dp.com/2/3d-printer-pid-tuning/. [Accessed 27 March 2024].

[25] J. O'Connell, “How to Fix Heat Creep (3D Printing): 5 Simple Solutions,” ALL3DP, 18 March 2024. [Online]. Available: https://all3dp.com/2/3d-printer-heat-creep/. [Accessed 27 March 2024].

[26] E3D, “Anatomy of a HotEnd,” 22 March 2021. [Online]. Available: https://e3d-online.com/blogs/news/anatomy-of-a-hotend?gad_source=1&gclid=CjwKCAjwh4-wBhB3EiwAeJsppEKIU2oYDraLT3uFlWusCbl2yeyiehQ_5US3UdWpK99dvXkXyle9WxoCfKEQAvD_BwE. [Accessed 27 March 2024].

[27] C. Duty, J. Failla, S. Kim, T. Smith, J. Lindahl and V. Kunc, “Z-Pinning approach for 3D printing mechanically isotropic materials,” *Additive Manufacturing,* vol. 27, no. 16, pp. 175-184, 2019.

[28] H. Bakhtiari, M. Aamir and M. Tolouei-Rad, “Effect of 3D Printing Parameters on the Fatigue Properties of Parts Manufactured by Fused Filament Fabrication: A Review,” *Applied Sciences,* vol. 13, no. 2, pp. 904-931, 2023.

[29] S. Wickramasinghe, T. Do and P. Tran, “FDM-Based 3D Printing of Polymer and Associated Composite: A Review on Mechanical Properties, Defects and Treatments,” *Polymers,* vol. 12, no. 7, pp. 1529-1571, 2020.

[30] J. S. a, P. S. b and J. Gunasekaran, “A review on the various processing parameters in FDM,” *Materials Today: Proceedings,* vol. 37, no. 2, pp. 509-514, 2021.

[31] The British Standards Institution, “BS ISO 6983-1:2009, Automation systems and integration. Numerical control of machines. Program format and definitions of address words. Data format for positioning, line motion and contouring control systems,” BSI Standards Limited, London, 2023.

[32] Team Xometry, “Slicer in 3D Printing: Definition, Features, and How it Works,” Xometry, 12 September 2022. [Online]. Available: https://www.xometry.com/resources/3d-printing/what-is-a-slicer-in-3d-printing/. [Accessed 22 March 2024].

[33] M. D. Monzón, Z. Ortega, A. Martínez and F. Ortega, “Standardization in additive manufacturing: activities carried out by international organizations and projects,” *The International Journal of Advanced Manufacturing Technology,* vol. 76, pp. 1111-1121, 2015.

[34] M. Szilvśi-Nagy and G. Mátyási, “Analysis of STL files,” *Mathematical and Computer Modelling,* vol. 38, no. 7, pp. 945-960, 2003.

[35] Markforged, “How to Create High Quality STL Files for 3D Prints,” 22 June 2021. [Online]. Available: https://markforged.com/resources/blog/how-to-create-high-quality-stl-files-for-3d-prints. [Accessed 27 March 2024].

[36] thinkyhead, “G0-G1 - Linear Move,” Marlin, 16 December 2016. [Online]. Available: https://marlinfw.org/docs/gcode/G000-G001.html. [Accessed 22 March 2024].

[37] J. Bryła and A. Martowicz, “Study on the Importance of a Slicer Selection for the 3D Printing Process Parameters via the Investigation of G-Code Readings,” *Machines,* vol. 9, no. 8, pp. 163-181, 2021.

[38] D. S. Chiniwar, H. Alva, V. R. Varada, M. Balichakra and S. Hiremath, “NVESTIGATION OF AUTOMATIC BED LEVELLING SYSTEM FOR FUSED DEPOSITION MODELLING 3D PRINTER MACHINE,” *International Journal of Modern Manufacturing Technologies,* vol. 14, no. 1, 2022.

[39] E. J. Kirkland, “Bilinear Interpolation,” in *Advanced Computing in Electron Microscopy*, vol. 2, Ithiaca, Springer, 2010, pp. 261-263.

[40] ABB, “Technical reference manual, RAPID Instructions, Functions and Data types,” 2010. [Online]. Available: https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf. [Accessed 25 March 2024].

[41] ABB, “Operating manual, Trouble shooting, IRC5,” 2010. [Online]. Available: https://library.e.abb.com/public/e6617595547fc6c2c1257cc5004451bd/Operating%20manual_Trouble%20shooting_3HAC020738-001_revK_en.pdf. [Accessed 25 March 2024].

[42] A. D'Souza, S. Vijayakumar and S. Schaal, “Learning inverse kinematics,” in *International Conference on Intelligent Robots and Systems*, Maui, 2001.

[43] F.-T. Cheng, T.-L. Hour, Y.-Y. Sun and T.-H. Chen, “Study and Resolution of Singularities for a 6-DOF PUMA Manipulator,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 27, no. 2, pp. 332-343, 1997.

[44] S. Zou, H. Liu, Y. Liu, J. Yao and H. Wu, “Singularity Analysis and Representation of 6DOF Parallel Robot Using Natural Coordinates,” *Journal of Robotics,* vol. 2021, 2021.

[45] Y. Nakamura and H. Hanafusa, “Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control,” *Journal of Dynamic Systems, Measurements and Control,* vol. 108, no. 3, pp. 163-171, 1986.

[46] Z.-H. Kang, C.-A. Cheng and H.-P. Huang, “A singularity handling algorithm based on operational space control for six-degree-of-freedom anthropomorphic manipulators,” *International Journal of Advanced Robotic Systems,* vol. 16, no. 3, 2019.

[47] I. Agustí-Juan and G. Habert, “Environmental Design Guidelines for Digital Fabrication,” *Journal of Cleaner Production,* vol. 142, no. 4, pp. 2780-2791, 2017.

[48] R. Horgan, “Cost of UK construction injuries soars to more than £16bn a year,” Construction News, 12 April 2022. [Online]. Available: https://www.constructionnews.co.uk/health-and-safety/cost-of-uk-construction-injuries-soars-to-more-than-16bn-a-year-12-04-2022/#:~:text=Injuries%20and%20ill%2Dhealth%20in,than%20businesses%20and%20government%20combined.. [Accessed 25 March 2024].

[49] C. B. Wendy Wilson, “Tackling the under-supply of,” UK Parliament, House of Commons Library, London, 2023.

[50] A. L. M. Tobi, S. A. Omar, Z. Yehia, S. Al-Ojaili, A. Hashim and O. Orhan, “Cost viability of 3D printed house in UK,” in *IOP Conference Series: Materials Science and Engineering*, Pahat, 2018.

[51] S. El-Sayegh, L. Romdhane and S. Manjikian, “A critical review of 3D printing in construction: benefits, challenges, and risks,” *Archives of Civil and Mechanical Engineering,* vol. 20, no. 34, 2020.

[52] S. Mostafavi and H. Bier, “Materially Informed Design to Robotic Production: A Robotic 3D Printing System for Informed Material Deposition,” in *Robotic Fabrication in Architecture, Art and Design 2016*, Cham, Springer, 2016, pp. 338-349.

[53] Z. Xu, T. Song, S. Guo, J. Peng, L. Zeng and M. Zhu, “Robotics technologies aided for 3D printing in construction: a review,” *The International Journal of Advanced Manufacturing Technology,* vol. 118, p. 3559–3574, 2022.

[54] Team Xometry, “3D Printing Technology for Construction,” Xometry, 18 May 2023. [Online]. Available: https://www.xometry.com/resources/3d-printing/3d-printing-in-construction/#:~:text=Overall%2C%203D%20printers%20in%20the,skill%20of%20the%20printer%20operator.. [Accessed 27 March 2024].

[55] SHoP, “Flotsam & Jetsam,” 06 January 2019. [Online]. Available: https://www.shoparc.com/projects/design-miami/. [Accessed 27 March 2024].

[56] D. W. Bao, X. Yan, R. Snooks and Y. M. Xie, “Bioinspired Generative Architectural Design Form-Finding and Advanced Robotic Fabrication Based on Structural Performance,” *Architectural Intelligence,* vol. 3, no. 1, pp. 147-160, 2020.

[57] R. Taylor, “Organ donation in England and the UK: Statistics and law changes,” UK Parliament, 06 December 2023. [Online]. Available: https://lordslibrary.parliament.uk/organ-donation-in-england-and-the-uk-statistics-and-law-changes/#:~:text=The%20NHS%20has%20stressed%20there,the%20end%20of%20March%202023.. [Accessed 26 March 2024].

[58] A. Parihar, V. Pandita, A. Kumar, D. S. Parihar, N. Puranik, T. Bajpai and R. Khan, “3D Printing: Advancement in Biogenerative Engineering to Combat Shortage of Organs and Bioapplicable Materials,” *Regenerative Engineering and Translational Medicine,* vol. 8, no. 2, p. 173–199, 2021.

[59] L. Budde, S. Ihler, S. Spindeldreier, T. Lücking, T. Meyer, EberhardBodenschatz and W.-H. Zimmermann, “A Six Degree of Freedom Extrusion Bioprinter,” *Current Directions in Biomedical Engineering,* vol. 8, no. 2, pp. 137-140, 2022.

[60] Z. Zhang, C. Wu, C. Dai, Q. Shi, G. Fang, D. Xie, Z. Xiangjie, Y.-J. Liu and C. Wang, “A multi-axis robot-based bioprinting system supporting natural cell function preservation and cardiac tissue fabrication,” *Bioactive Materials,* vol. 18, no. 11, pp. 138-150, 2022.

[61] N. E. Fedorovich, J. Alblas, W. E. Hennink, F. C. Oner and W. J. A. Dhert, “Organ printing: the future of bone regeneration?,” *Trends in Biotechnology,* vol. 29, no. 12, pp. 601-606, 2011.

[62] S. H. Jariwala, G. S. Lewis, Z. J. Bushman, J. H. Adair and H. J. Donahue, “3D Printing of Personalized Artificial Bone Scaffolds,” *3D PRINTING,* vol. 2, no. 2, pp. 56-64, 2015.

[63] E. F. Morgan, G. U. Unnikrisnan and A. I. Hussein, “Bone Mechanical Properties in Healthy and Diseased States,” *Annual Review of Biomedical Engineering,* vol. 20, pp. 119-143, 2018.

[64] K. Ma, T. Zhao, L. Yang, P. Wang and X. Wang, “Application of robotic-assisted in situ 3D printing in cartilage regeneration with HAMA hydrogel: An in vivo study,” *Journal of Advanced Research,* vol. 23, no. 11, pp. 123-132, 2020.

[65] P. Tang, X. Zhao, H. Shi, B. Hu, J. Ding, B. Yang and W. Xu, “A review of multi-axis additive manufacturing: Potential, opportunity and challenge,” *Additive Manufacturing,* vol. 83, 2024.

[66] H. Zeng, M. Feng, J. Zhuang, R. Cai, Y. Xie and J. Li, “3D Printing And Free-form Surface Coating Based on 6-DOF Robot,” in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dali, 2019.

[67] D. Ahlers, F. Wasserfall, N. Hendrich and J. Zhang, “3D Printing of Nonplanar Layers for Smooth Surface Generation,” in *IEEE 15th International Conference on Automation Science and Engineering (CASE)*, Hamburg, 2019.

[68] I. Mitropoulou, M. Bernhard and B. Dillenburger, “Nonplanar 3D Printing of Bifurcating Forms,” *3D Printing and Additive Manufacturing,* vol. 9, no. 3, 2022.

[69] T. Morita, S. Watanabe and S. Sasaki, “Multiaxis printing method for bent tubular structured gels in support bath for achieving high dimension and shape accuracy,” *Precision Engineering ,* vol. 79, no. 10, pp. 109-118, 2023.

[70] Y. Li, K. Xu, X. Liu, M. Yang, J. Gao and P. Maropoulos, “Stress-oriented 3D printing path optimization based on image processing algorithms for reinforced load-bearing parts,” *CIRP Annals - Manufacturing Technology,* vol. 70, pp. 195-198, 2021.

[71] E. Sales, T.-H. Kwok and Y. Chen, “Function-aware slicing using principal stress line for toolpath planning in additive manufacturing,” *Journal of Manufacturing Processes ,* vol. 64, pp. 1420-1433, 2021.

[72] A. Hancock, “Additive Manufacturing Market Size, Share & Trends Analysis Report by 2030,” Vantage Market Research, 2024.

[73] Precedence Research, “Computer Numerical Control (CNC) Machine Market,” 24 August 2023. [Online]. Available: https://www.precedenceresearch.com/computer-numerical-control-machine-market. [Accessed 28 March 2024].

[74] A. A. Erumban, “Lifetimes of machinery and equipment: Evidence from Dutch manufacturing,” *Review of Income and Wealth,* vol. 54, no. 2, pp. 237-268, 2008.

[75] I. Pearse, “Design and Characterisation of a Robotic 3D Printing System,” University of Bath, Bath, 2024.

[76] British Standards Institution, “BS 4391:1972, Recommendations for metric basic sizes for metal wire,” British Standards Institution, London, 1972.

[77] British Standards Institution, “BS 9550:1989, Specification for capability approval procedures for d.c. and low frequency connector cable assemblies and wiring harnesses: generic data,” British Standards Institution, London, 1989.

[78] NASA, “NASA Workmanship Standards,” 28 June 2002. [Online]. Available: https://workmanship.nasa.gov/lib/insp/2%20books/links/sections/Appendix.html. [Accessed 18 March 2024].

[79] L. M. Sonneborn and F. S. V. Vleck, “The Bang-Bang Principle for Linear Control Systems,” *Journal on Control and Optimization,* vol. 2, no. 2, pp. 151-159, 1964.

[80] D. A. Anderegg, H. A. Bryant, D. C. Ruffin, S. M. S. Jr., J. J. Fallon, E. L. Gilmer and M. J. Bortner, “In-situ monitoring of polymer flow temperature and pressure in extrusion based additive manufacturing,” *Additive Manufacturing,* vol. 26, no. 8, pp. 76-83, 2019.

[81] S. Campbell, “MAKE AN ARDUINO TEMPERATURE SENSOR (THERMISTOR TUTORIAL),” Circuit Basics, 18 November 2015. [Online]. Available: https://www.circuitbasics.com/arduino-thermistor-temperature-sensor-tutorial/. [Accessed 18 March 2024].

[82] D. Aguirre, “GCode-to-ABB,” GitHub, 22 February 2019. [Online]. Available: https://github.com/DAguirreAg/GCode-to-ABB/blob/master/GCode_to_Robtargets.py. [Accessed 10 April 2024].

[83] drag&bot Help, “ABB Controller Setup,” 26 February 2020. [Online]. Available: https://help.dragandbot.com/hardware/robots/abb/01_irc5_setup.html. [Accessed 19 March 2024].

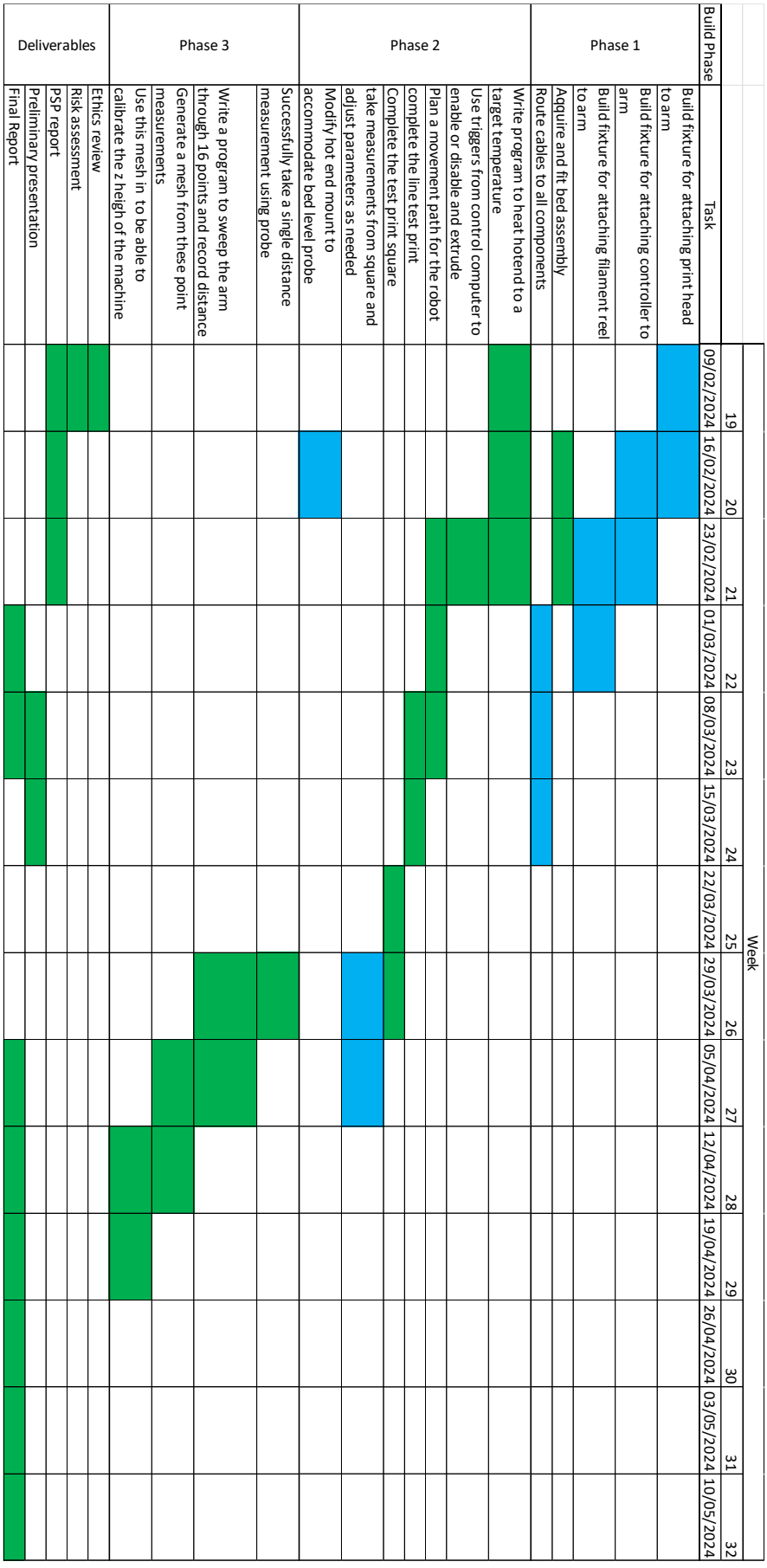
[84] iDig3Dprinting, “XYZ 20mm Calibration Cube,” Thingiverse, 19 January 2016. [Online]. Available: https://www.thingiverse.com/thing:1278865. [Accessed 10 April 2024].

[85] P. Cruse, “Hemera XS Cooling Duct,” Printables, 04 September 2022. [Online]. Available: https://www.printables.com/model/249117-hemera-xs-cooling-duct. [Accessed 10 April 2024].

[86] 3DBenchy, “Features,” 2020. [Online]. Available: https://3dbenchy.com/features/. [Accessed 20 March 2024].

[87] jmcharg, “BLTouch-Tester,” Github, 30 January 2018. [Online]. Available: https://github.com/jmcharg/BLTouch-Tester/blob/master/BLTouch_Tester_v1.0.ino. [Accessed 29 April 2024].

6. Appendices



Appendix 1 - Gantt chart showing proposed project timeline.

```
// 16/04/2024
// Louis Huygens
// ReadThermistor program
// Based on program by Scott Campbell, 18/11/2015

// Program reads voltage across resistor, compares against the known 5V input, and
uses this to calculate the voltage dropped over the thermistor
// Thermal coefficients then used to relate resistance to temperature
// Results scaled appropriately
//Declare variables
int ThermistorPin = 0;
int Vo;
float R1 = 10000;
float logR2, R2, T, Tc, Tf;
float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;
float Tcc;
float Time;
float CurrentTime;

//Start timer
void setup() {
Serial.begin(9600);
Time = millis();
}

void loop() {

  //Calculate temperature
  Vo = analogRead(ThermistorPin);
  R2 = R1 * (1023.0 / (float)Vo - 1.0);
  logR2 = log(R2);
  T = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2));
  Tc = T - 273.15;
  Tf = (Tc * 9.0)/ 5.0 + 32.0;
  //Scale
  Tcc = (-8.95 * Tc) + 745.79;

  //Count time elapsed
  CurrentTime = millis() - Time;

  //Record temperatures for 160s period
  if (CurrentTime < 160000){
    Serial.println(Tcc);
  }

  //Limit number of data points
  delay(50);
}
```

Appendix 2 - Arduino program for reading temperature from the hot end thermistor.

```
// 16/04/2024
// Louis Huygens
// runStepper10 Program

// Program runs stepper motor for 10s with specified pulse delay

//Declare variables
float time;
float startTime;

void setup() {

  Serial.begin(9600);
  pinMode(5, OUTPUT);
  digitalWrite(5, LOW);

  //Start timer
  time = 10 * 1000;
  startTime = millis();

}

void loop() {
  float elapsedTime = millis() - startTime;

  //Pulse at constant rate for 10s
  if(elapsedTime < time){
    digitalWrite(5, LOW);
    delay(8.72);  //12.787
    digitalWrite(5, HIGH);

  }
  else {
    digitalWrite(5, LOW);
  }
}
```

Appendix 3 - Program for running stepper motor with a specific pulse delay for 10s for calibration.

```
# Originally Created by: Daniel Aguirre
# Date: 22/02/2019
# Modified by Louis Huygens
# Date:29/02/2024

# Original file took a GCode from a laser cutter slicer and converted it to
# Rapid, plotting a 2D path for drawing on paper, with a pen as the tool

# Modifications allow for increases in z height for 3D printing. changes have
# been made to the parsing command and the generation of the output files.
# The modified program outputs coordinates in X,Y,Z, and writes code for speed
# and retraction commands.



# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import sys, getopt
import scipy.io

# GLOBAL VARIABLES
inputfile = "3DBenchy_0.2mm_PLA_NEPTUNE2S_1h33m.gcode"
outputfile_robtargets = "Benchy_robtargets.txt"
outputfile_moveLs = "Benchy_moveLs.txt"

rotation = "[0,0.7071068,-0.7071068,0], "
conf = "[0,0,0,1]"

G90 = True

# Program variables
positions = pd.DataFrame([[0.0,0.0,0.0]], columns=["x","y","z"])
Speed = pd.DataFrame([5], columns=["f"])
Retraction = pd.DataFrame([1], columns=["e"])
robtargets = []
moveLs = []
NumArrays = []
dz = 0.0
de = 1


# Extract GCode command specific information
def parseCommand(command, G90):
    global positions
    global Speed
    global Retraction
    global dz
    global de

    temp = command.split()
```

```
if len(temp) == 0:
    temp = [";"]

if temp[0]=="G0" or temp[0]=="G1":

    x = 0.0
    y = 0.0
    z = 0.0
    f = 0.0
    e = 0.0
    dx = 0.0
    dy = 0.0
    df = 0.0

    for comp in temp:
        #   look for speed commands
        if comp.startswith("F"):
            df = round(float(comp[1:]) / 100)
            dx = positions.iloc[-1].x
            dy = positions.iloc[-1].y
        #  look for X coordinates
        if comp.startswith("X"):
            dx = float(comp[1:])
            dz = positions.iloc[-1].z
            df = Speed.iloc[-1].f
        #  look for Y coordinates
        if comp.startswith("Y"):
            dy = float(comp[1:])
            dz = positions.iloc[-1].z
            df = Speed.iloc[-1].f
        if comp.startswith("Z"):
            dx = positions.iloc[-1].x
            dy = positions.iloc[-1].y
            df = Speed.iloc[-1].f
            dz = float(comp[1:])
        #   look for retraction commands
        if comp.startswith("E-.8"):
            de = 1
            dx = positions.iloc[-1].x
            dy = positions.iloc[-1].y
        #   look for detraction commands
        if comp.startswith("E.8"):
            de = 0
            dx = positions.iloc[-1].x
            dy = positions.iloc[-1].y

        x = dx
        y = dy
        z = dz
        f = df
        e = de
```

```
        #Write to arrays
        newPosition = pd.DataFrame([[x,y,z]], columns=["x","y","z"])
        positions = pd.concat([positions, newPosition])

        newSpeed = pd.DataFrame([f], columns=["f"])
        Speed = pd.concat([Speed, newSpeed])

        newRetraction = pd.DataFrame([e], columns=["e"])
        Retraction = pd.concat([Retraction, newRetraction])

# Writes the Robtarget points into a file
def writeRobtarget(i, position):
    x = position.x
    y = position.y
    z = round(position.z + 0.1, 2)

    string1 = "CONST robtarget "
    string2 = "d" + str(i)
    string3 = ":= [[" + str(x) + "," + str(y) + "," + str(z) + "], " + rotation + conf + ", [ 9E+9,9E+9, 9E9, 9E9, 9E9,
9E9]];"
    string4 = "\n"
    robtarget = string1 + string2 + string3 + string4
    robtargets.append(robtarget)

# Writes the GCode points into a file
def writeNumArray(i, position):
    x = position.x
    y = position.y
    z = position.z
    string3 = "[" + str(x) + "," + str(y) + "," + str(z) + "], "
    string4 = "\n"
    NumArray = string3 + string4
    NumArrays.append(NumArray)

# Writes the ABB move commands into a file
def writeMoveL(i, Speed, Retraction):
    f = int(Speed.f)
    r = round(Retraction.e)
    if r == 1:
        fb = 0;
    if r == 0:
        fb = f;
    fb = format(fb, '07b')

    f64 = fb[::-1][6]
    f32 = fb[::-1][5]
    f16 = fb[::-1][4]
    f8 = fb[::-1][3]
    f4 = fb[::-1][2]
    f2 = fb[::-1][1]
    f1 = fb[::-1][0]

    string1 = "MoveL "
```

```
    string2 = "d" + str(i)
    string3 = ",v" + str(f) +", z0, Hemera\WObj:=PrintBed;"
    string4 = "\n"
    string5 = "setDO D652_10_DO10, " + str(f64) + "; "
    string6 = "setDO D652_10_DO11, " + str(f32) + "; "
    string7 = "setDO D652_10_DO12, " + str(f16) + "; "
    string8 = "setDO D652_10_DO13, " + str(f8) + "; "
    string9 = "setDO D652_10_DO14, " + str(f4) + "; "
    string10 = "setDO D652_10_DO15, " + str(f2) + "; "
    string11 = "setDO D652_10_DO16, " + str(f1) + "; "
    string12 = "setDO D652_10_DO9, " + str(r) + "; "

    moveL =  string5 + string6 + string7 + string8 + string9 + string10 + string11 + string12 + string4 + string1 +
string2 + string3 + string4
    moveLs.append(moveL)

def plotPath(proyection="2d"):
    x = np.array(positions.x, dtype=pd.Series)
    y = np.array(positions.y, dtype=pd.Series)
    z = np.array(positions.z, dtype=pd.Series)

    x0 = np.array([1,])
    y0 = np.array([1,])

    fig = plt.figure()

    if (proyection == "3d"):
        ax = Axes3D(fig)
        ax.plot(x,y,z)

    else:
        ax = fig.add_subplot(111)
        ax.plot(x,y,"red")
        ax.scatter(x,y)

    plt.show()


################### MAIN ###################
def main(argv):

    global inputfile, outputfile_robtargets, outputfile_moveLs, rotation, conf

    try:
        opts, args = getopt.getopt(argv,"hi:o:r:c:",["help","ifile=","ofile="])
    except getopt.GetoptError:
        print('Error')
        sys.exit(2)


    for opt, arg in opts:
        if opt in ("-h", "--help"):
            print("Usage: GCode_to_Robtargets [-h | -i <inputfile> -o <outputfile>] ")
```

```
        print('Options and arguments:')
        print("-h     : Print this help message and exit")
        print("-i arg : Input the file to be converted into ABB instructions (also --ifile)")
        print("-o arg : Output filename containing the ABB instructions (also --ofile)")
        print("-r arg : Specify the rotation of the robtargets (also --rot). Default: [-1, 0, 0, 0]")
        print("-c arg : Specify the axis configuration of the robtargets (also --conf). Default: [-1, 0, 1, 0]")
        sys.exit()

    elif opt in ("-i", "--ifile"):
        inputfile = arg

    elif opt in ("-o", "--ofile"):
        outputfile_robtargets = arg + "_robtargets.txt"
        outputfile_moveLs = arg + "_moveLs.txt"

    elif opt in ("-r", "--rot"):
        rotation = arg

    elif opt in ("-c", "--conf"):
        conf = arg


# Check if Input file has been defined
if inputfile == None:
     print("Inputfile not defined")
     sys.exit(2)

# Load GCode and obtain XYZ coordinates
file = open(inputfile,"r")
with open(inputfile,"r") as file:
    line = file.readline()
    lineNumberCount = 1
    while line:
        print("Line: " + str(lineNumberCount))
        line = file.readline()
        parseCommand(line, G90)
        lineNumberCount += 1

# Write Robtargets and MoveL to a txt file
for i in range(0, positions.shape[0]-1):
    position = positions.iloc[i]
    speed = Speed.iloc[i]
    retraction = Retraction.iloc[i]
    writeRobtarget(i, position)
    writeMoveL(i, speed, retraction)

with open(outputfile_robtargets,"w") as file:
    for line in robtargets:
        file.writelines(line)

with open(outputfile_moveLs,"w") as file:
    for line in moveLs:
        file.writelines(line)
```

```
    print("Conversion finished")

    # Plot expected result
    plotPath()


if __name__ == "__main__":
  main(sys.argv[1:])
```

Appendix 4 - Python program for converting GCode to Rapid movement and target commands.

```
MODULE Module1
   CONST robtarget Zero_10:=[[0,0,200],[0,0.7071068,-
0.7071068,0],[0,0,0,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

   VAR speeddata v0 := [ 0, 500, 5000, 1000 ];
   VAR speeddata v1 := [ 1, 500, 5000, 1000 ];
   VAR speeddata v2 := [ 2, 500, 5000, 1000 ];
   VAR speeddata v3 := [ 3, 500, 5000, 1000 ];
   VAR speeddata v4 := [ 4, 500, 5000, 1000 ];
   VAR speeddata v5 := [ 5, 500, 5000, 1000 ];
   VAR speeddata v6 := [ 6, 500, 5000, 1000 ];
   VAR speeddata v7 := [ 7, 500, 5000, 1000 ];
   VAR speeddata v8 := [ 8, 500, 5000, 1000 ];
   VAR speeddata v9 := [ 9, 500, 5000, 1000 ];
   VAR speeddata v10 := [ 10, 500, 5000, 1000 ];
   VAR speeddata v11 := [ 11, 500, 5000, 1000 ];
   VAR speeddata v12 := [ 12, 500, 5000, 1000 ];
   VAR speeddata v13 := [ 13, 500, 5000, 1000 ];
   VAR speeddata v14 := [ 14, 500, 5000, 1000 ];
   VAR speeddata v15 := [ 15, 500, 5000, 1000 ];
   VAR speeddata v16 := [ 16, 500, 5000, 1000 ];
   VAR speeddata v17 := [ 17, 500, 5000, 1000 ];
   VAR speeddata v18 := [ 18, 500, 5000, 1000 ];
   VAR speeddata v19 := [ 19, 500, 5000, 1000 ];
   VAR speeddata v20 := [ 20, 500, 5000, 1000 ];
   VAR speeddata v21 := [ 21, 500, 5000, 1000 ];
   VAR speeddata v22 := [ 22, 500, 5000, 1000 ];
   VAR speeddata v23 := [ 23, 500, 5000, 1000 ];
   VAR speeddata v24 := [ 24, 500, 5000, 1000 ];
   VAR speeddata v25 := [ 25, 500, 5000, 1000 ];
   VAR speeddata v26 := [ 26, 500, 5000, 1000 ];
   VAR speeddata v27 := [ 27, 500, 5000, 1000 ];
   VAR speeddata v28 := [ 28, 500, 5000, 1000 ];
   VAR speeddata v29 := [ 29, 500, 5000, 1000 ];
   VAR speeddata v30 := [ 30, 500, 5000, 1000 ];
   VAR speeddata v31 := [ 31, 500, 5000, 1000 ];
   VAR speeddata v32 := [ 32, 500, 5000, 1000 ];
   VAR speeddata v33 := [ 33, 500, 5000, 1000 ];
   VAR speeddata v34 := [ 34, 500, 5000, 1000 ];
   VAR speeddata v35 := [ 35, 500, 5000, 1000 ];
   VAR speeddata v36 := [ 36, 500, 5000, 1000 ];
   VAR speeddata v37 := [ 37, 500, 5000, 1000 ];
   VAR speeddata v38 := [ 38, 500, 5000, 1000 ];
   VAR speeddata v39 := [ 39, 500, 5000, 1000 ];
   VAR speeddata v40 := [ 40, 500, 5000, 1000 ];
   VAR speeddata v41 := [ 41, 500, 5000, 1000 ];
   VAR speeddata v42 := [ 42, 500, 5000, 1000 ];
   VAR speeddata v43 := [ 43, 500, 5000, 1000 ];
   VAR speeddata v44 := [ 44, 500, 5000, 1000 ];
   VAR speeddata v45 := [ 45, 500, 5000, 1000 ];
   VAR speeddata v46 := [ 46, 500, 5000, 1000 ];
   VAR speeddata v47 := [ 47, 500, 5000, 1000 ];
   VAR speeddata v48 := [ 48, 500, 5000, 1000 ];
```

```
VAR speeddata v49 := [ 49, 500, 5000, 1000 ];
VAR speeddata v50 := [ 50, 500, 5000, 1000 ];
VAR speeddata v51 := [ 51, 500, 5000, 1000 ];
VAR speeddata v52 := [ 52, 500, 5000, 1000 ];
VAR speeddata v53 := [ 53, 500, 5000, 1000 ];
VAR speeddata v54 := [ 54, 500, 5000, 1000 ];
VAR speeddata v55 := [ 55, 500, 5000, 1000 ];
VAR speeddata v56 := [ 56, 500, 5000, 1000 ];
VAR speeddata v57 := [ 57, 500, 5000, 1000 ];
VAR speeddata v58 := [ 58, 500, 5000, 1000 ];
VAR speeddata v59 := [ 59, 500, 5000, 1000 ];
VAR speeddata v60 := [ 60, 500, 5000, 1000 ];
VAR speeddata v61 := [ 61, 500, 5000, 1000 ];
VAR speeddata v62 := [ 62, 500, 5000, 1000 ];
VAR speeddata v63 := [ 63, 500, 5000, 1000 ];
VAR speeddata v64 := [ 64, 500, 5000, 1000 ];
VAR speeddata v65 := [ 65, 500, 5000, 1000 ];
VAR speeddata v66 := [ 66, 500, 5000, 1000 ];
VAR speeddata v67 := [ 67, 500, 5000, 1000 ];
VAR speeddata v68 := [ 68, 500, 5000, 1000 ];
VAR speeddata v69 := [ 69, 500, 5000, 1000 ];
VAR speeddata v70 := [ 70, 500, 5000, 1000 ];
VAR speeddata v71 := [ 71, 500, 5000, 1000 ];
VAR speeddata v72 := [ 72, 500, 5000, 1000 ];
VAR speeddata v73 := [ 73, 500, 5000, 1000 ];
VAR speeddata v74 := [ 74, 500, 5000, 1000 ];
VAR speeddata v75 := [ 75, 500, 5000, 1000 ];
VAR speeddata v76 := [ 76, 500, 5000, 1000 ];
VAR speeddata v77 := [ 77, 500, 5000, 1000 ];
VAR speeddata v78 := [ 78, 500, 5000, 1000 ];
VAR speeddata v79 := [ 79, 500, 5000, 1000 ];
VAR speeddata v80 := [ 80, 500, 5000, 1000 ];
VAR speeddata v81 := [ 81, 500, 5000, 1000 ];
VAR speeddata v82 := [ 82, 500, 5000, 1000 ];
VAR speeddata v83 := [ 83, 500, 5000, 1000 ];
VAR speeddata v84 := [ 84, 500, 5000, 1000 ];
VAR speeddata v85 := [ 85, 500, 5000, 1000 ];
VAR speeddata v86 := [ 86, 500, 5000, 1000 ];
VAR speeddata v87 := [ 87, 500, 5000, 1000 ];
VAR speeddata v88 := [ 88, 500, 5000, 1000 ];
VAR speeddata v89 := [ 89, 500, 5000, 1000 ];
VAR speeddata v90 := [ 90, 500, 5000, 1000 ];
VAR speeddata v91 := [ 91, 500, 5000, 1000 ];
VAR speeddata v92 := [ 92, 500, 5000, 1000 ];
VAR speeddata v93 := [ 93, 500, 5000, 1000 ];
VAR speeddata v94 := [ 94, 500, 5000, 1000 ];
VAR speeddata v95 := [ 95, 500, 5000, 1000 ];
VAR speeddata v96 := [ 96, 500, 5000, 1000 ];
VAR speeddata v97 := [ 97, 500, 5000, 1000 ];
VAR speeddata v98 := [ 98, 500, 5000, 1000 ];
VAR speeddata v99 := [ 99, 500, 5000, 1000 ];
VAR speeddata v100 := [ 100, 500, 5000, 1000 ];
VAR speeddata v101 := [ 101, 500, 5000, 1000 ];
```

```
VAR speeddata v102 := [ 102, 500, 5000, 1000 ];
VAR speeddata v103 := [ 103, 500, 5000, 1000 ];
VAR speeddata v104 := [ 104, 500, 5000, 1000 ];
VAR speeddata v105 := [ 105, 500, 5000, 1000 ];
VAR speeddata v106 := [ 106, 500, 5000, 1000 ];
VAR speeddata v107 := [ 107, 500, 5000, 1000 ];
VAR speeddata v108 := [ 108, 500, 5000, 1000 ];
VAR speeddata v109 := [ 109, 500, 5000, 1000 ];
VAR speeddata v110 := [ 110, 500, 5000, 1000 ];
VAR speeddata v111 := [ 111, 500, 5000, 1000 ];
VAR speeddata v112 := [ 112, 500, 5000, 1000 ];
VAR speeddata v113 := [ 113, 500, 5000, 1000 ];
VAR speeddata v114 := [ 114, 500, 5000, 1000 ];
VAR speeddata v115 := [ 115, 500, 5000, 1000 ];
VAR speeddata v116 := [ 116, 500, 5000, 1000 ];
VAR speeddata v117 := [ 117, 500, 5000, 1000 ];
VAR speeddata v118 := [ 118, 500, 5000, 1000 ];
VAR speeddata v119 := [ 119, 500, 5000, 1000 ];
VAR speeddata v120 := [ 120, 500, 5000, 1000 ];
VAR speeddata v121 := [ 121, 500, 5000, 1000 ];
VAR speeddata v122 := [ 122, 500, 5000, 1000 ];
VAR speeddata v123 := [ 123, 500, 5000, 1000 ];
VAR speeddata v124 := [ 124, 500, 5000, 1000 ];
VAR speeddata v125 := [ 125, 500, 5000, 1000 ];
VAR speeddata v126 := [ 126, 500, 5000, 1000 ];
VAR speeddata v127 := [ 127, 500, 5000, 1000 ];
VAR speeddata v128 := [ 128, 500, 5000, 1000 ];
VAR speeddata v129 := [ 129, 500, 5000, 1000 ];
VAR speeddata v130 := [ 130, 500, 5000, 1000 ];
VAR speeddata v131 := [ 131, 500, 5000, 1000 ];
VAR speeddata v132 := [ 132, 500, 5000, 1000 ];
VAR speeddata v133 := [ 133, 500, 5000, 1000 ];
VAR speeddata v134 := [ 134, 500, 5000, 1000 ];
VAR speeddata v135 := [ 135, 500, 5000, 1000 ];
VAR speeddata v136 := [ 136, 500, 5000, 1000 ];
VAR speeddata v137 := [ 137, 500, 5000, 1000 ];
VAR speeddata v138 := [ 138, 500, 5000, 1000 ];
VAR speeddata v139 := [ 139, 500, 5000, 1000 ];
VAR speeddata v140 := [ 140, 500, 5000, 1000 ];
VAR speeddata v141 := [ 141, 500, 5000, 1000 ];
VAR speeddata v142 := [ 142, 500, 5000, 1000 ];
VAR speeddata v143 := [ 143, 500, 5000, 1000 ];
VAR speeddata v144 := [ 144, 500, 5000, 1000 ];
VAR speeddata v145 := [ 145, 500, 5000, 1000 ];
VAR speeddata v146 := [ 146, 500, 5000, 1000 ];
VAR speeddata v147 := [ 147, 500, 5000, 1000 ];
VAR speeddata v148 := [ 148, 500, 5000, 1000 ];
VAR speeddata v149 := [ 149, 500, 5000, 1000 ];
VAR speeddata v150 := [ 150, 500, 5000, 1000 ];
VAR speeddata testSpeed := [ 20, 50, 5000, 100 ];

VAR loadsession load1;
VAR loadsession load2;
```

```
    VAR loadsession load3;
    VAR loadsession load4;
    VAR loadsession load5;
    VAR loadsession load6;
    VAR loadsession load7;
    VAR loadsession load8;
    VAR loadsession load9;
    VAR loadsession load10;
    VAR loadsession load11;
    VAR loadsession load12;
    VAR loadsession load13;
    VAR loadsession load14;

PROC Main()

        setDO D652_10_DO10, 0; setDO D652_10_DO11, 0; setDO D652_10_DO12, 0; setDO D652_10_DO13,
0; setDO D652_10_DO14, 0; setDO D652_10_DO15, 0; setDO D652_10_DO16, 0; setDO D652_10_DO9, 0;
        MoveL Zero_10,v50,z0,Hemera\WObj:=PrintBed;


!       %"FourCornerMod:FourCorner"%;
!       %"LinesMod:Lines"%;
!       %"SquareMod:Square"%;
!       %"LinesAdjMod:LinesAdj"%;


        Load \Dynamic, "HOME:/BenchyMod1.MOD";
        %"Benchy1"%;
        UnLoad "HOME:/BenchyMod1.MOD";

        Load \Dynamic, "HOME:/BenchyMod2.MOD";
        %"Benchy2"%;
        UnLoad "HOME:/BenchyMod2.MOD";

        Load \Dynamic, "HOME:/BenchyMod3.MOD";
        %"Benchy3"%;
        UnLoad "HOME:/BenchyMod3.MOD";

        Load \Dynamic, "HOME:/BenchyMod4.MOD";
        %"Benchy4"%;
        UnLoad "HOME:/BenchyMod4.MOD";

        Load \Dynamic, "HOME:/BenchyMod5.MOD";
        %"Benchy5"%;
        UnLoad "HOME:/BenchyMod5.MOD";

        Load \Dynamic, "HOME:/BenchyMod6.MOD";
        %"Benchy6"%;
        UnLoad "HOME:/BenchyMod6.MOD";

        Load \Dynamic, "HOME:/BenchyMod7.MOD";
        %"Benchy7"%;
        UnLoad "HOME:/BenchyMod7.MOD";
```

```
    Load \Dynamic, "HOME:/BenchyMod8.MOD";
    %"Benchy8"%;
    UnLoad "HOME:/BenchyMod8.MOD";

    Load \Dynamic, "HOME:/BenchyMod9.MOD";
    %"Benchy9"%;
    UnLoad "HOME:/BenchyMod9.MOD";

    Load \Dynamic, "HOME:/BenchyMod10.MOD";
    %"Benchy10"%;
    UnLoad "HOME:/BenchyMod10.MOD";

    Load \Dynamic, "HOME:/BenchyMod11.MOD";
    %"Benchy11"%;
    UnLoad "HOME:/BenchyMod11.MOD";

    Load \Dynamic, "HOME:/BenchyMod12.MOD";
    %"Benchy12"%;
    UnLoad "HOME:/BenchyMod12.MOD";

    Load \Dynamic, "HOME:/BenchyMod13.MOD";
    %"Benchy13"%;
    UnLoad "HOME:/BenchyMod13.MOD";

    Load \Dynamic, "HOME:/BenchyMod14.MOD";
    %"Benchy14"%;
    UnLoad "HOME:/BenchyMod14.MOD";

    Load \Dynamic, "HOME:/BenchyMod15.MOD";
    %"Benchy15"%;
    UnLoad "HOME:/BenchyMod15.MOD";



    setDO D652_10_DO10, 0; setDO D652_10_DO11, 0; setDO D652_10_DO12, 0; setDO D652_10_DO13,
0; setDO D652_10_DO14, 0; setDO D652_10_DO15, 0; setDO D652_10_DO16, 0; setDO D652_10_DO9, 0;
    MoveL Zero_10,v50,z0,Hemera\WObj:=PrintBed;
  ENDPROC

ENDMODULE
```

Appendix 5 - Main Rapid module uploaded onto the robot controller.

```
// 16/04/2024
// Louis Huygens
// BinaryPrintSpeed program

// Program reads inputs on pins 1 to 12
// Converts binary input from pins to integer value
// sets the speed of the stepper motor to this value

//Declare variables
int A,B,C,D,E,F,G,H;
int Bin = 0b00000000;
unsigned long previousMillis = 0;
float Delay = 50;

//Assign IO
void setup() {
  Serial.begin(9600);
  pinMode(4, OUTPUT); //Direction Out Pin
  pinMode(5, OUTPUT); //Pulse Out Pin
  pinMode(6, INPUT_PULLUP); //1 In Pin
  pinMode(7, INPUT_PULLUP); //2 In Pin
  pinMode(8, INPUT_PULLUP); //4 In Pin
  pinMode(9, INPUT_PULLUP); //8 In Pin
  pinMode(10, INPUT_PULLUP);  //16 In Pin
  pinMode(11, INPUT_PULLUP);  //32 In Pin
  pinMode(12, INPUT_PULLUP);  //64 In Pin
  pinMode(13, OUTPUT);  //Direction In Pin
}

void loop() {

  //Read input value
  Bin = InToBinary();

  //Convert speed in mm/s to a signal delay
  Delay = 1 / (Bin * 0.9 * 1.638) * 1000;
  Serial.println(Bin);
  Serial.println(Delay);

  //Avoid zero speed values
  if (Bin == 0){
    Bin = Bin + 0.001;
  }

  //Check if time elapsed greater than delay, if so pulse stepper
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= Delay) {
    previousMillis = currentMillis;
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
```

```
    digitalWrite(5, LOW);
  }
}

int InToBinary()
{

  int Sum = 0;
  if (digitalRead(6) == LOW){
    Sum += 1;}
  if (digitalRead(7) == LOW){
    Sum += 2;}
  if (digitalRead(8) == LOW){
    Sum += 4;}
  if (digitalRead(9) == LOW){
    Sum += 8;}
  if (digitalRead(10) == LOW){
    Sum += 16;}
  if (digitalRead(11) == LOW){
    Sum += 32;}
  if (digitalRead(12) == LOW){
    Sum += 64;}

  return Sum;
}
```

Appendix 6 - Arduino program for reading a binary input, converting to an integer value, and setting the stepper motor to the inputted speed.

```
//Louis Huygens
//28/02/2024


// Program for running the print head of 6DoF robotic 3D printer
// Takes a demand speed in the form of binary input on pins 6-12
// Ouputs speed to stepper driver in the form of a pulsed signal
//  performs retraction when direction pin changes bit

//Declare variables
int A, B, C, D, E, F, G, H;
int Bin;
unsigned long previousMillis = 0;
float Delay = 50;
bool State = 0;
bool lastState = 0;

//Assign IO
void setup() {
  Serial.begin(9600);
  pinMode(13, INPUT_PULLUP);
  pinMode(4, OUTPUT);          //Direction Out Pin
  pinMode(5, OUTPUT);          //Pulse Out Pin
  pinMode(6, INPUT_PULLUP);    //1 In Pin
  pinMode(7, INPUT_PULLUP);    //2 In Pin
  pinMode(8, INPUT_PULLUP);    //4 In Pin
  pinMode(9, INPUT_PULLUP);    //8 In Pin
  pinMode(10, INPUT_PULLUP);   //16 In Pin
  pinMode(11, INPUT_PULLUP);   //32 In Pin
  pinMode(12, INPUT_PULLUP);   //64 In Pin
}

void loop() {
  State = digitalRead(13);

  //Perform retraction upon change of pin state
  if (State != lastState) {
    if (digitalRead(13) == LOW) {
      digitalWrite(30, HIGH);
      for (int i = 0; i <= 3; i++) {
        digitalWrite(5, HIGH);
        digitalWrite(5, LOW);
        delay(17.44);
      }
    }
  }

  //Convert to binary
  Bin = InToBinary();
  //Calculate pulse delay
```

```
  Delay = 1 / (Bin * 0.9 * 1.638) * 1000;

  //Account for infinite pulse
  if (Bin == 0) {
    Bin = Bin + 0.001;
  }


  //Output pulse signal
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= Delay) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
    digitalWrite(5, LOW);
  }

  lastState = State;
}

//Calculate input in binary
int InToBinary() {
  int Sum = 0;

  if (digitalRead(6) == LOW) {
    Sum += 1;
  }
  if (digitalRead(7) == LOW) {
    Sum += 2;
  }
  if (digitalRead(8) == LOW) {
    Sum += 4;
  }
  if (digitalRead(9) == LOW) {
    Sum += 8;
  }
  if (digitalRead(10) == LOW) {
    Sum += 16;
  }
  if (digitalRead(11) == LOW) {
    Sum += 32;
  }
  if (digitalRead(12) == LOW) {
    Sum += 64;
  }
  return (Sum);
}
```

Appendix 7 - Arduino program to run the print head during printing operation.

```
/*
  16/04/2024
  Louis Huygens
  BLTouchBedLevel program

  Modified from code by jmcharg
  https://github.com/jmcharg/BLTouch-Tester/blob/master/BLTouch_Tester_v1.0.ino
  BLTouch Test sensor program
  Connect 5V to Red
  Connect Gnd to Brown
  Connect Pin 9 to Orange / Yellow
  Connect Pin 2 to White.

  Serial port to 9600 and dont send CR or LF

  Send 1 to Pin Down
  Send 2 to Pin Up
  Send 3 to Test
  Send 4 to Reset

*/

#include <Servo.h>

Servo myservo;

int val;
int incomingByte = 0;

const byte BLTouchPin = 2;  // Connect the white wire from the BLTouch to this pin
const byte BLTouchControl = 3;   // Connect the orange wire from the BLTouch sensor
to this pin
volatile byte state = LOW;
volatile byte lastState = LOW;

void setup() {
  Serial.begin(9600);
  myservo.attach(BLTouchControl);
  pinMode(2, INPUT_PULLUP);
  myservo.write(60);
  pinMode(A5, OUTPUT);
  digitalWrite(A5, LOW);
}

void loop() {
  //Set pin position down
  myservo.write(10);

  //If servo touched, lift pin, set output and reset
  if (digitalRead(2) == HIGH) {
```

```
    myservo.write(90);
    Serial.println("Resetting");
    digitalWrite(A5, HIGH);
    delay(1000);
    digitalWrite(A5, LOW);
    myservo.write(10);
  }
}
```

Appendix 8 - Arduino program for controlling the BLTouch bed levelling sensor.

```
MODULE BLMod
        VAR robtarget sp1;
        VAR robtarget sp2;
        VAR robtarget sp3;
        VAR robtarget sp4;
        VAR robtarget sp5;
        VAR robtarget sp6;
        VAR robtarget sp7;
        VAR robtarget sp8;
        VAR robtarget sp9;
        VAR robtarget sp10;
        VAR robtarget sp11;
        VAR robtarget sp12;
        VAR robtarget sp13;
        VAR robtarget sp14;
        VAR robtarget sp15;
        VAR robtarget sp16;
        VAR robtarget sp17;
        VAR robtarget sp18;
        VAR robtarget sp19;
        VAR robtarget sp20;
        VAR robtarget sp21;
        VAR robtarget sp22;
        VAR robtarget sp23;
        VAR robtarget sp24;
        VAR robtarget sp25;


        CONST robtarget c0:= [[0.0,0.0,150], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c1:= [[0.0,0.0,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c2:= [[15,15,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c3:= [[65,15,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c4:= [[115,15,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c5:= [[165,15,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c6:= [[215,15,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c7:= [[15,65,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c8:= [[65,65,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c9:= [[115,65,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c10:= [[165,65,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c11:= [[215,65,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
```

```
        CONST robtarget c12:= [[15,115,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c13:= [[65,115,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c14:= [[115,115,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c15:= [[165,115,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c16:= [[215,115,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c17:= [[15,165,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c18:= [[65,165,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c19:= [[115,165,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c20:= [[165,165,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c21:= [[215,165,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c22:= [[15,215,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c23:= [[65,215,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c24:= [[115,215,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c25:= [[165,215,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];
        CONST robtarget c26:= [[215,215,0.0], [0,0.701057385,-0.701057385,0], [0,0,0,1], [ 9E+9,9E+9, 9E9,
9E9, 9E9, 9E9]];

    PROC BL()
        MoveL c0,v50, fine, Hemera\WObj:=PrintBed;
                MoveL OFFS(c1,0,0,20),v50, fine, Hemera\WObj:=PrintBed;

                MoveL OFFS(c2,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
                SearchL \Stop, D652_10_DI10, sp1, c2, v1, Hemera\WObj:=PrintBed;
                MoveL OFFS(c2,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

                MoveL OFFS(c3,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
                SearchL \Stop, D652_10_DI10, sp2, c3, v1, Hemera\WObj:=PrintBed;
                MoveL OFFS(c3,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

                MoveL OFFS(c4,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
                SearchL \Stop, D652_10_DI10, sp3, c4, v1, Hemera\WObj:=PrintBed;
                MoveL OFFS(c4,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

                MoveL OFFS(c5,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
                SearchL \Stop, D652_10_DI10, sp4, c5, v1, Hemera\WObj:=PrintBed;
                MoveL OFFS(c5,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

                MoveL OFFS(c6,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
                SearchL \Stop, D652_10_DI10, sp5, c6, v1, Hemera\WObj:=PrintBed;
```

```
MoveL OFFS(c6,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c7,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp6, c7, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c7,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c8,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp7, c8, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c8,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c9,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp8, c9, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c9,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c10,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp9, c10, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c10,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c11,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp10, c11, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c11,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c12,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp11, c12, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c12,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c13,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp12, c13, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c13,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c14,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp13, c14, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c14,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c15,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp14, c15, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c15,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c16,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp15, c16, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c16,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c17,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp16, c17, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c17,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c18,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp17, c18, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c18,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

MoveL OFFS(c19,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
SearchL \Stop, D652_10_DI10, sp18, c19, v1, Hemera\WObj:=PrintBed;
MoveL OFFS(c19,0,0,20),v10, fine, Hemera\WObj:=PrintBed;
```

```
        MoveL OFFS(c20,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
        SearchL \Stop, D652_10_DI10, sp19, c20, v1, Hemera\WObj:=PrintBed;
        MoveL OFFS(c20,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

        MoveL OFFS(c21,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
        SearchL \Stop, D652_10_DI10, sp20, c21, v1, Hemera\WObj:=PrintBed;
        MoveL OFFS(c21,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

        MoveL OFFS(c22,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
        SearchL \Stop, D652_10_DI10, sp21, c22, v1, Hemera\WObj:=PrintBed;
        MoveL OFFS(c22,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

        MoveL OFFS(c23,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
        SearchL \Stop, D652_10_DI10, sp22, c23, v1, Hemera\WObj:=PrintBed;
        MoveL OFFS(c23,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

        MoveL OFFS(c24,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
        SearchL \Stop, D652_10_DI10, sp23, c24, v1, Hemera\WObj:=PrintBed;
        MoveL OFFS(c24,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

        MoveL OFFS(c25,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
        SearchL \Stop, D652_10_DI10, sp24, c25, v1, Hemera\WObj:=PrintBed;
        MoveL OFFS(c25,0,0,20),v10, fine, Hemera\WObj:=PrintBed;

        MoveL OFFS(c26,0,0,20),v50, fine, Hemera\WObj:=PrintBed;
        SearchL \Stop, D652_10_DI10, sp25, c26, v1, Hemera\WObj:=PrintBed;
        MoveL OFFS(c26,0,0,20),v10, fine, Hemera\WObj:=PrintBed;


        MoveL c0,v50, fine, Hemera\WObj:=PrintBed;

    ENDPROC
ENDMODULE
```

Appendix 9 - Rapid code for probing the bed.

```
# Originally Created by: Daniel Aguirre
# Date: 22/02/2019
# Modified by Louis Huygens
# Date:29/02/2024

# Original file took a GCode from a laser cutter slicer and converted it to
# Rapid, plotting a 2D path for drawing on paper, with a pen as the tool

# Modifications allow for increases in z height for 3D printing. changes have
# been made to the parsing command and the generation of the output files.
# The modified program outputs coordinates in X,Y,Z, and writes code for speed
# and retraction commands.



# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import sys, getopt
import scipy.io
import math

# GLOBAL VARIABLES
inputfile = "3DBenchy_0.2mm_PLA_NEPTUNE2S_1h33m.gcode"
outputfile_robtargets = "Benchy_robtargets.txt"
outputfile_moveLs = "Benchy_moveLs.txt"

rotation = "[0,0.7071068,-0.7071068,0], "
conf = "[0,0,0,1]"

G90 = True

# Program variables
positions = pd.DataFrame([[0.0,0.0,0.0]], columns=["x","y","z"])
Speed = pd.DataFrame([5], columns=["f"])
Retraction = pd.DataFrame([1], columns=["e"])
robtargets = []
moveLs = []
NumArrays = []
dz = 0.0
de = 1

#Bed probe coordinates
p1 = (0, 0, 2.57608)
p2 = (50, 0, 2.86989)
p3 = (100, 0, 2.94418)
p4 = (150, 0, 2.85357)
p5 = (200, 0, 2.56837)
p6 = (0, 50, 2.55952)
p7 = (50, 50, 2.82228)
p8 = (100, 50, 2.9139)
```

```
p9 = (150, 50, 2.80468)
p10 = (200, 50, 2.50553)
p11 = (0, 100, 2.52471)
p12 = (50, 100, 2.82434)
p13 = (100, 100, 2.89199)
p14 = (150, 100, 2.81364)
p15 = (200, 100, 2.54148)
p16 = (0, 150, 2.53342)
p17 = (50, 150, 2.79708)
p18 = (100, 150, 2.88318)
p19 = (150, 150, 2.81295)
p20 = (200, 150, 2.61035)
p21 = (0, 200, 2.55512)
p22 = (50, 200, 2.798938)
p23 = (100, 200, 2.82164)
p24 = (150, 200, 2.75293)
p25 = (200, 200, 2.54133)

#Bed matrices
Q1 = [p1, p2, p6, p7]
Q2 = [p2, p3, p7, p8]
Q3 = [p3, p4, p8, p9]
Q4 = [p4, p5, p9, p10]
Q5 = [p6, p7, p11, p12]
Q6 = [p7, p8, p12, p13]
Q7 = [p8, p9, p13, p14]
Q8 = [p9, p10, p14, p15]
Q9 = [p11, p12, p16, p17]
Q10 = [p12, p13, p17, p18]
Q11 = [p13, p14, p18, p19]
Q12 = [p14, p15, p19, p20]
Q13 = [p16, p17, p21, p22]
Q14 = [p17, p18, p22, p23]
Q15 = [p18, p19, p23, p24]
Q16 = [p19, p20, p24, p25]


#Credit to Amir, https://math.stackexchange.com/questions/2975109/how-to-convert-euler-angles-to-
quaternions-and-get-the-same-euler-angles-back-fr

#quaternion conversion
def euler_to_quaternion(roll, pitch, yaw):
    qx = np.sin(roll/2) * np.cos(pitch/2) * np.cos(yaw/2) - np.cos(roll/2) * np.sin(pitch/2) * np.sin(yaw/2)
    qy = np.cos(roll/2) * np.sin(pitch/2) * np.cos(yaw/2) + np.sin(roll/2) * np.cos(pitch/2) * np.sin(yaw/2)
    qz = np.cos(roll/2) * np.cos(pitch/2) * np.sin(yaw/2) - np.sin(roll/2) * np.sin(pitch/2) * np.cos(yaw/2)
    qw = np.cos(roll/2) * np.cos(pitch/2) * np.cos(yaw/2) + np.sin(roll/2) * np.sin(pitch/2) * np.sin(yaw/2)

    return [round(qw, 7), round(qx, 7), round(qy, 7), round(qz, 7)]

#Bilinear interpolation function
def bilinear_interpolation(x, y, points):
    '''Interpolate (x,y) from values associated with four points.
```

```
    The four points are a list of four triplets:  (x, y, value).
    The four points can be in any order.  They should form a rectangle.

        >>> bilinear_interpolation(12, 5.5,
        ...                        [(10, 4, 100),
        ...                         (20, 4, 200),
        ...                         (10, 6, 150),
        ...                         (20, 6, 300)])
        165.0

    '''
    # See formula at:  http://en.wikipedia.org/wiki/Bilinear_interpolation

    points = sorted(points)               # order points by x, then by y
    (x1, y1, q11), (_x1, y2, q12), (x2, _y1, q21), (_x2, _y2, q22) = points

    if x1 != _x1 or x2 != _x2 or y1 != _y1 or y2 != _y2:
        raise ValueError('points do not form a rectangle')
    if not x1 <= x <= x2 or not y1 <= y <= y2:
        raise ValueError('(x, y) not within the rectangle')

    return (q11 * (x2 - x) * (y2 - y) +
            q21 * (x - x1) * (y2 - y) +
            q12 * (x2 - x) * (y - y1) +
            q22 * (x - x1) * (y - y1)
           ) / ((x2 - x1) * (y2 - y1) + 0.0)

#Perform interpolation from an input matrix
def local_interp(n):

    matrix = []
    xMax = n[1][0]
    xMin = n[0][0]
    yMax = n[2][1]
    yMin = n[0][1]
    for i in range (yMin, yMax+1):
        b = []
        for j in range(xMin, xMax+1):
            b.append(round(bilinear_interpolation(j, i, n),3))
        matrix.append(b)

    return np.flipud(np.array(matrix))

#combine matrices
def genBedMatrix():
    #     np.savetxt('Q1_Matrix.txt', Q1m, fmt='%f')


    Q1m = local_interp(Q1)
    Q2m = local_interp(Q2)
    Q3m = local_interp(Q3)
    Q4m = local_interp(Q4)
    Q5m = local_interp(Q5)
```

```
    Q6m = local_interp(Q6)
    Q7m = local_interp(Q7)
    Q8m = local_interp(Q8)
    Q9m = local_interp(Q9)
    Q10m = local_interp(Q10)
    Q11m = local_interp(Q11)
    Q12m = local_interp(Q12)
    Q13m = local_interp(Q13)
    Q14m = local_interp(Q14)
    Q15m = local_interp(Q15)
    Q16m = local_interp(Q16)

    Q2m = np.delete(Q2m, 0, 1)
    Q3m = np.delete(Q3m, 0, 1)
    Q4m = np.delete(Q4m, 0, 1)

    Q5m = np.delete(Q5m, 0, 0)
    Q6m = np.delete(Q6m, 0, 1)
    Q6m = np.delete(Q6m, 0, 0)
    Q7m = np.delete(Q7m, 0, 1)
    Q7m = np.delete(Q7m, 0, 0)
    Q8m = np.delete(Q8m, 0, 1)
    Q8m = np.delete(Q8m, 0, 0)

    Q9m = np.delete(Q9m, 0, 0)
    Q10m = np.delete(Q10m, 0, 1)
    Q10m = np.delete(Q10m, 0, 0)
    Q11m = np.delete(Q11m, 0, 1)
    Q11m = np.delete(Q11m, 0, 0)
    Q12m = np.delete(Q12m, 0, 1)
    Q12m = np.delete(Q12m, 0, 0)

    Q13m = np.delete(Q13m, 0, 0)
    Q14m = np.delete(Q14m, 0, 1)
    Q14m = np.delete(Q14m, 0, 0)
    Q15m = np.delete(Q15m, 0, 1)
    Q15m = np.delete(Q15m, 0, 0)
    Q16m = np.delete(Q16m, 0, 1)
    Q16m = np.delete(Q16m, 0, 0)


    R1 = np.hstack((Q1m, Q2m, Q3m, Q4m))
    R2 = np.hstack((Q5m, Q6m, Q7m, Q8m))
    R3 = np.hstack((Q9m, Q10m, Q11m, Q12m))
    R4 = np.hstack((Q13m, Q14m, Q15m, Q16m))

    bedMatrix = np.vstack((R4, R3, R2, R1))
    bedMatrix = np.flipud(np.array(bedMatrix))
    return bedMatrix


# Extract GCode command specific information
def parseCommand(command, G90):
```

```
    global positions
    global Speed
    global Retraction
    global dz
    global de

    temp = command.split()

    if len(temp) == 0:
        temp = [";"]

    # if temp[0] ==";LAYER_CHANGE":

    #     dz = round(dz + 0.2, 2)

    if temp[0]=="G0" or temp[0]=="G1":

        x = 0.0
        y = 0.0
        z = 0.0
        f = 0.0
        e = 0.0
        dx = 0.0
        dy = 0.0
        df = 0.0

        for comp in temp:
            #   look for speed commands
            if comp.startswith("F"):
                df = round(float(comp[1:]) / 100)
                dx = positions.iloc[-1].x
                dy = positions.iloc[-1].y
            #  look for X coordinates
            if comp.startswith("X"):
                dx = float(comp[1:])
                dz = positions.iloc[-1].z
                df = Speed.iloc[-1].f
            #  look for Y coordinates
            if comp.startswith("Y"):
                dy = float(comp[1:])
                dz = positions.iloc[-1].z
                df = Speed.iloc[-1].f
            if comp.startswith("Z"):
                dx = positions.iloc[-1].x
                dy = positions.iloc[-1].y
                df = Speed.iloc[-1].f
                dz = float(comp[1:])
            if comp.startswith("E-.8"):
                de = 1
                dx = positions.iloc[-1].x
                dy = positions.iloc[-1].y
            if comp.startswith("E.8"):
                de = 0
```

```
            dx = positions.iloc[-1].x
            dy = positions.iloc[-1].y

        x = dx
        y = dy
        z = dz
        f = df
        e = de

    newPosition = pd.DataFrame([[x,y,z]], columns=["x","y","z"])
    positions = pd.concat([positions, newPosition])

    newSpeed = pd.DataFrame([f], columns=["f"])
    Speed = pd.concat([Speed, newSpeed])

    newRetraction = pd.DataFrame([e], columns=["e"])
    Retraction = pd.concat([Retraction, newRetraction])

# Writes the Robtarget points into a file
def writeRobtarget(i, position, matrix):
    x = position.x
    y = position.y

    xr = round(x)
    yr = round(y)

    #If outside of probed region
    if xr < 17.5 or xr > 217.5 or yr < 17.5 or yr >217.5:
        zr = 2.70
        Thetax = 0
        Thetay = 0
    #Set height to height from matrix, calculate head angle
    else:
        zr = matrix[round(yr-17.5)][round(xr-17.5)]
        dx1 = matrix[round(yr-18.5)][round(xr-18.5)] - matrix[round(yr-18.5)][round(xr-16.5)]
        dx2 = matrix[round(yr-16.5)][round(xr-18.5)] - matrix[round(yr-16.5)][round(xr-16.5)]
        dy1 = matrix[round(yr-16.5)][round(xr-18.5)] - matrix[round(yr-18.5)][round(xr-18.5)]
        dy2 = matrix[round(yr-16.5)][round(xr-16.5)] - matrix[round(yr-18.5)][round(xr-16.5)]

        Thetax = math.atan((dx1+dx2)/4)
        Thetay = math.atan((dy1+dy2)/4)

    quart = euler_to_quaternion(-math.pi - Thetax, Thetay, -math.pi/2)

    z = round(position.z + zr - 2.85, 2)

    string1 = "CONST robtarget "
    string2 = "a" + str(i)
    string3 = ":= [[" + str(x) + "," + str(y) + "," + str(z) + "], " + str(quart) + "," + conf + ", [ 9E+9,9E+9, 9E9, 9E9,
9E9, 9E9]];"
    string4 = "\n"
    robtarget = string1 + string2 + string3 + string4
    robtargets.append(robtarget)
```

```
# Writes the GCode points into a file
def writeNumArray(i, position):
    x = position.x
    y = position.y
    z = position.z
    string3 = "[" + str(x) + "," + str(y) + "," + str(z) + "], "
    string4 = "\n"
    NumArray = string3 + string4
    NumArrays.append(NumArray)

# Writes the ABB move commands into a file
def writeMoveL(i, Speed, Retraction):
    f = int(Speed.f)
    r = round(Retraction.e)
    if r == 1:
        fb = 0;
    if r == 0:
        fb = f;
    fb = format(fb, '07b')

    f64 = fb[::-1][6]
    f32 = fb[::-1][5]
    f16 = fb[::-1][4]
    f8 = fb[::-1][3]
    f4 = fb[::-1][2]
    f2 = fb[::-1][1]
    f1 = fb[::-1][0]

    string1 = "MoveL "
    string2 = "a" + str(i)
    string3 = ",v" + str(f) +", z0, Hemera\WObj:=PrintBed;"
    string4 = "\n"
    string5 = "setDO D652_10_DO10, " + str(f64) + "; "
    string6 = "setDO D652_10_DO11, " + str(f32) + "; "
    string7 = "setDO D652_10_DO12, " + str(f16) + "; "
    string8 = "setDO D652_10_DO13, " + str(f8) + "; "
    string9 = "setDO D652_10_DO14, " + str(f4) + "; "
    string10 = "setDO D652_10_DO15, " + str(f2) + "; "
    string11 = "setDO D652_10_DO16, " + str(f1) + "; "
    string12 = "setDO D652_10_DO9, " + str(r) + "; "

    moveL =  string5 + string6 + string7 + string8 + string9 + string10 + string11 + string12 + string4 + string1 +
string2 + string3 + string4
    moveLs.append(moveL)

def plotPath(proyection="2d"):
    x = np.array(positions.x, dtype=pd.Series)
    y = np.array(positions.y, dtype=pd.Series)
    z = np.array(positions.z, dtype=pd.Series)

    x0 = np.array([1,])
    y0 = np.array([1,])
```

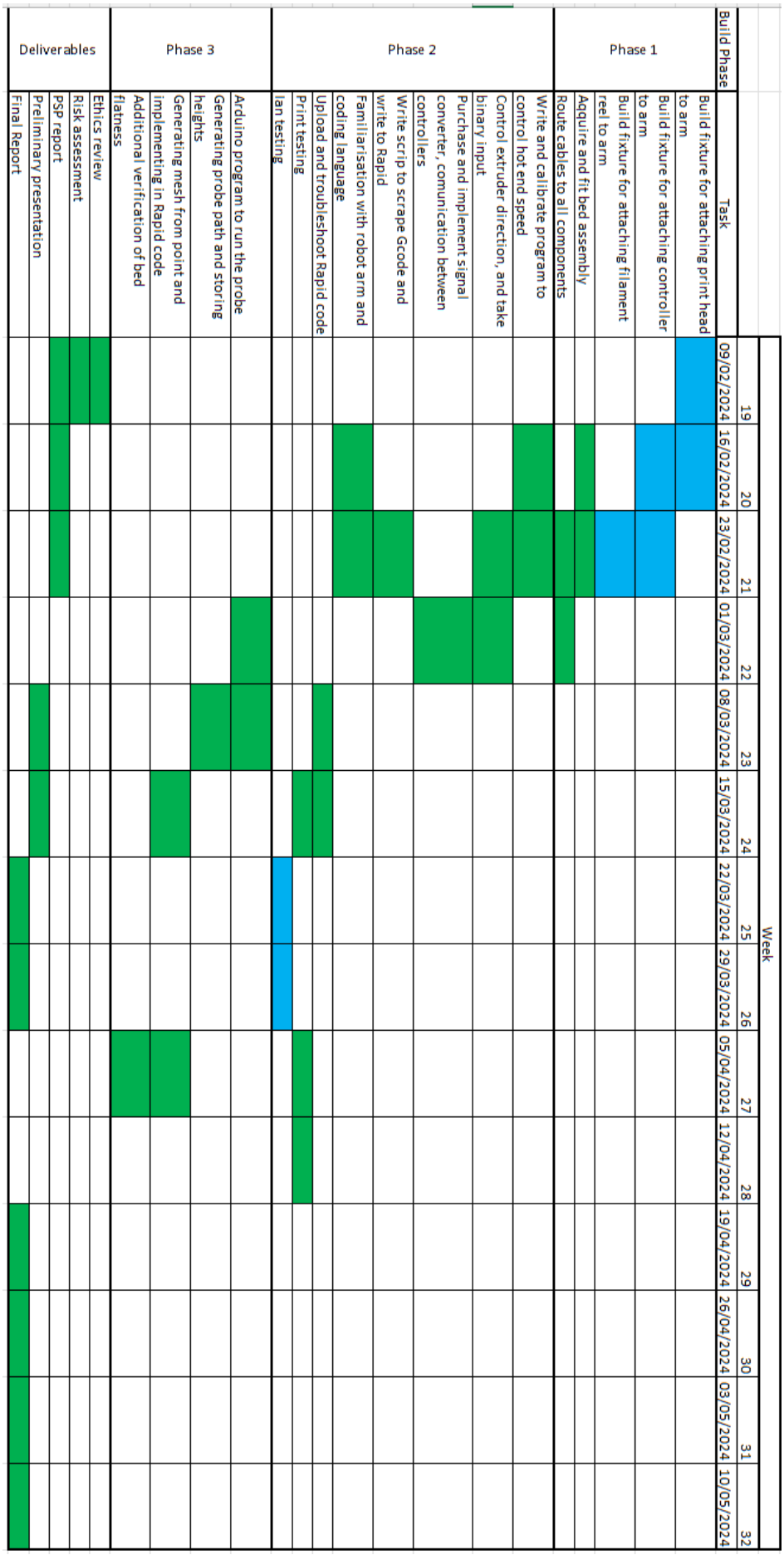
```
    fig = plt.figure()

    if (proyection == "3d"):
        ax = Axes3D(fig)
        ax.plot(x,y,z)

    else:
        ax = fig.add_subplot(111)
        ax.plot(x,y,"red")
        ax.scatter(x,y)

    plt.show()


################### MAIN ###################
def main(argv):

    global inputfile, outputfile_robtargets, outputfile_moveLs, rotation, conf

    try:
        opts, args = getopt.getopt(argv,"hi:o:r:c:",["help","ifile=","ofile="])
    except getopt.GetoptError:
        print('Error')
        sys.exit(2)

    bedMatrix = genBedMatrix()


    for opt, arg in opts:
        if opt in ("-h", "--help"):
            print("Usage: GCode_to_Robtargets [-h | -i <inputfile> -o <outputfile>] ")
            print('Options and arguments:')
            print("-h     : Print this help message and exit")
            print("-i arg : Input the file to be converted into ABB instructions (also --ifile)")
            print("-o arg : Output filename containing the ABB instructions (also --ofile)")
            print("-r arg : Specify the rotation of the robtargets (also --rot). Default: [-1, 0, 0, 0]")
            print("-c arg : Specify the axis configuration of the robtargets (also --conf). Default: [-1, 0, 1, 0]")
            sys.exit()

        elif opt in ("-i", "--ifile"):
            inputfile = arg

        elif opt in ("-o", "--ofile"):
            outputfile_robtargets = arg + "_robtargets.txt"
            outputfile_moveLs = arg + "_moveLs.txt"

        elif opt in ("-r", "--rot"):
            rotation = arg

        elif opt in ("-c", "--conf"):
            conf = arg
```

```
    # Check if Input file has been defined
    if inputfile == None:
        print("Inputfile not defined")
        sys.exit(2)

    # Load GCode and obtain XYZ coordinates
    file = open(inputfile,"r")
    with open(inputfile,"r") as file:
        line = file.readline()
        lineNumberCount = 1
        while line:
            print("Line: " + str(lineNumberCount))
            line = file.readline()
            parseCommand(line, G90)
            lineNumberCount += 1

    # Write Robtargets and MoveL to a txt file
    for i in range(0, positions.shape[0]-1):
        position = positions.iloc[i]
        speed = Speed.iloc[i]
        retraction = Retraction.iloc[i]
        writeRobtarget(i, position, bedMatrix)
        writeMoveL(i, speed, retraction)

    with open(outputfile_robtargets,"w") as file:
        for line in robtargets:
            file.writelines(line)

    with open(outputfile_moveLs,"w") as file:
        for line in moveLs:
            file.writelines(line)

    print("Conversion finished")

    # Plot expected result
    plotPath()


if __name__ == "__main__":
   main(sys.argv[1:])
```

Appendix 10 - Adapted GCode to Rapid python script which accounts for bed height and angle.

Ref	Description	Link	Qty	Cost (inc VAT)	Sum
1	Print Bed Gantry	https://www.amazon.co.uk/Zeberoxyz-Carriage-190mmx190mmx3-5mm-7-5x7-5x0-14inch-190x190x3-5mm-Lightweight/dp/B08YYDRLFG/ref=sr_1_13_sspa?crid=2Y5DINQRSZ6QC&keywords=Y%2BCarriage%2BPlate%2BKit&qid=1707215015&s=industrial&sprefix=y%2Bcarriage%2Bplate%2Bkit%2Cindustrial%2C64&sr=1-13-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9tdGY&th=1	1	£24.99	£24.99
2	Bed Levelling Springs	https://www.amazon.co.uk/Super-Print-Heatbed-Leveling-Silicone/dp/B09VC8NG14/ref=asc_df_B09VC8NG14/?tag=googshopuk-21&linkCode=df0&hvadid=606569762767&hvpos=&hvnetw=g&hvrand=5757287507413439825&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1006502&hvtargid=pla-1871531207976&mcid=a68122855ffa37c69a63ea474e70e33f&th=1	1	£13.98	£13.98
3	Print Bed	https://www.amazon.co.uk/Original-Aluminum-Heatbed-Platform-Ender-3/dp/B094MKMV5T/ref=sr_1_10?crid=4MSNUCCNAVIB&keywords=3d%2Bprinter%2Bheated%2Bbed&qid=1707322582&sprefix=3d%2Bprinter%2Bheated%2Bbed%2Caps%2C182&sr=8-10&th=1	1	£29.99	£29.99
4	Optocoupler	https://www.amazon.co.uk/8-Channel-Optocoupler-Isolated-PLC-PNP-Converter/dp/B0B6NWLQWG	1	£11.99	£11.99
5	Black Hook Up Wire	https://uk.rs-online.com/web/p/hook-up-wire/1681559?searchId=5e30ffd0-6ee5-4c01-b17c-457cb6edb797&gb=s	2	£14.90	£29.80
6	Red Hook Up Wire	https://uk.rs-online.com/web/p/hook-up-wire/1681571?searchId=b1acd7f8-0cf5-4548-8e17-f5f9cc3eea33&gb=s	2	£14.90	£29.80
7	M3 Square Nuts	https://uk.rs-online.com/web/p/square-nuts/0837262?searchId=0ac39a31-f629-44f6-bb87-4a7c57868353&gb=s	1	£6.06	£6.06
8	M3x12 Bolts	https://uk.rs-online.com/web/p/machine-screws/4733518?searchId=48e5950f-0f00-4b00-a06b-709dddb1391d&gb=s	1	£9.35	£9.35
9	Multimeter	https://uk.rs-online.com/web/p/multimeters/1611625?searchId=81f07484-6890-4ba4-a07e-9f813f9306ec&gb=s	1	£32.10	£32.10
10	Spirit Level	https://uk.rs-online.com/web/p/spirit-levels-inclinometers/7243721?gb=s	1	£16.44	£16.44
11	5015 12V blower fan	https://www.digikey.co.uk/en/products/detail/sunon-fans/MF50151V2-1B00U-A99/15996444?s=N4IgTCBcDa4KwE4C0BZAYnADARjtgamEtgEKaYCqSAggsgHIAilAugL5A	1	£6.05	£6.05
12	Relay	https://uk.rs-online.com/web/p/power-motor-robotics-development-tools/1845098	1	£7.55	£7.55
					£218.10

Appendix 11 - Expenditure of the project.



Appendix 12 - Revised Gantt chart reflective of actual time allocation.