POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Developing a Custom Printhead Controller to Print Large-Scale Parts Using Industrial Motion Systems

PAUL GREGORIO

Département de génie mécanique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie mécanique

Mai 2022

Developing a Custom Printhead Controller to Print Large-Scale Parts Using Industrial Motion Systems

DEDICATION

To my parents and sister, who have always encouraged me to do what I love and pushed me to do my best

ACKNOWLEDGEMENTS

I would like to thank Daniel Therriault for his helpful guidance and support during my time in the LM2. He has been a great source of inspiration, motivating his team to produce high-quality work while remaining humble, honest, and kind.

I would also like to thank Rouhollah Farahani, who keeps the laboratory running smoothly in the most hectic of times and ensures all students have what they need.

I would like to thank the NSERC and FRQNT for providing me with master’s scholarships, offering me the financial stability to focus on my research.

I would especially like to thank Abraham Bherer-Constant and Jean-François Chauvette, without whom I certainly would not have been able to achieve my master’s goal of printing on industrial motion systems. Their creativity and programming expertise were fundamental to this project’s success.

I would like to thank the team from Safran whose insight and advice were invaluable throughout my project, particularly Nicola Piccirelli and Sebastien Pautard.

I would like to warmly thank all my LM2 colleagues for making the laboratory a great place to work over the past years. I have the pleasure of calling you friends.

I would like to thank my parents and sister, who have always supported me in everything that I do.

Finally, I would like to thank my two cats Boris and Bambi for keeping me company as I wrote this thesis. You have stayed by my side even in the most difficult of times.

RÉSUMÉ

Le dépôt de fil fondu (FFF, de *fused filament fabrication*) pourrait devenir une puissante méthode de fabrication, mais cette technologie de fabrication additive (FA) doit d’abord surpasser certaines limites, dont sa mise à l’échelle et la qualité des pièces produites. Dans ce projet, nous avons adressé ce problème de taille en développant un contrôleur de tête d’impression personnalisé permettant de produire de grandes pièces par FFF sur des robots industriels.

L’objectif principal du projet était de créer un contrôleur universel pouvant être connecté à différents robots munis de têtes d’impression à haut débit. Le contrôleur a été conçu pour communiquer avec deux robots industriels spécifiques : un grand système à portique d'Aerotech et un bras robotique à six axes Fanuc M-20iB/25. Les deux systèmes sont installés dans le Laboratoire de mécanique multi-échelle (LM2) de Polytechnique Montréal. Une carte mère Duet 3 6HC est au cœur du contrôleur de tête d'impression, qui peut contrôler plusieurs outils à la fois et est compatible avec un large éventail de composants que l'on trouve couramment dans les têtes d'impression FFF. Nous avons choisi les autres composants internes du contrôleur selon les exigences de deux têtes d'impression à haut débit : les extrudeuses Typhoon et Pulsar de Dyze Design.

Pour utiliser le contrôleur de tête d'impression avec les robots industriels du LM2, nous avons développé une méthodologie d'impression adaptée. Pour ce faire, nous avons utilisé une méthode de contrôle d'extrusion personnalisée, implémenté des protocoles de communication, modifié le firmware de la carte mère Duet et développé des post-processeurs dans RoboDK, un logiciel de programmation hors-ligne pour robots industriels. Nous avons ensuite imprimé diverses pièces à l'aide de plusieurs têtes d'impression, dont l'extrudeuse Typhoon et la version bêta de l'extrudeuse Pulsar, afin de démontrer la polyvalence du contrôleur. Avec l'extrudeuse Typhoon et le robot Fanuc, nous avons produit deux grandes pièces: un vase en spirale mesurant 400 mm de hauteur et pesant 640 g (imprimé à 250 % de sa taille originale), et un modèle #3DBenchy de 240 mm de long et 636 g (à 400 % d'échelle). Les temps d'impression des deux pièces étaient d'environ 3h et 3h20m, respectivement. Pour le vase, nous avons estimé qu'il faudrait jusqu'à 18h42 pour produire la même pièce en utilisant une tête d’impression conventionnelle couramment utilisée dans les imprimantes FFF.

Malheureusement, la méthode de contrôle d'extrusion que nous avons mise en place a abouti à une qualité d’impression moyenne dont les effets étaient particulièrement observables sur le #3DBenchy. Ce mémoire comprend quelques recommandations pour améliorer cette situation à l'avenir. Une fois ces problèmes résolus, la flexibilité et la modularité du contrôleur de tête d'impression devraient en faire un outil de recherche particulièrement utile. Les chercheurs du LM2 pourront l'utiliser pour explorer de nouvelles idées reliées à la fabrication additive à grande échelle, ainsi que d'autres sujets d’actualité tels que la FFF multi-axes et la FA de composites haute performance.

ABSTRACT

Fused filament fabrication (FFF) could become a powerful manufacturing method if the technology can scale up and produce higher quality parts. In this project, we addressed FFF’s scale limitation by developing a custom printhead controller to produce large parts on industrial motion systems.

The project’s main objective was to create a universal controller that could easily be connected to different motion systems to print with high-flowrate, extrusion-based toolheads. We designed the controller to communicate with two specific motion systems: a large, gantry-based system from Aerotech and a Fanuc M-20iB/25 six-axis robotic arm. Both systems are installed in the Laboratory for Multiscale Mechanics (LM2) at Polytechnique Montreal. A Duet 3 Mainboard 6HC drives the printhead controller, which can control multiple toolheads at a time and is compatible with a wide range of components commonly found in FFF printheads. We selected the controller’s other internal components based on the requirements of two high-flowrate printheads: the Typhoon and Pulsar extruders from Dyze Design.

To use the printhead controller on the LM2 motion systems we developed a custom printing methodology. This required using an in-house extrusion control method, implementing communication protocols, modifying the Duet mainboard’s firmware, and developing post-processors in RoboDK, an offline programming software for industrial robots. We then printed various parts using the Typhoon extruder, the beta version of the Pulsar extruder, and several other printheads to demonstrate the controller’s versatility. Using the Typhoon extruder on the Fanuc six-axis robot, we produced two large-scale parts: a 400mm tall, 640g spiral vase (printed at 250% its original size), and a 240mm long, 636g #3DBenchy benchmark part (400% scale). Print times were approximately 3h and 3h20m respectively. For the vase, we estimated it would take up to 18h42m to produce the same part using standard toolheads typically used on FFF printers.

Unfortunately, the extrusion control method we implemented meant print quality on the #3DBenchy in particular was substandard. This thesis includes a few recommendations as to how to improve on this in the future. Once these issues have been resolved, the printhead controller’s flexibility and modularity should help it become a useful research tool. Researchers will be able to use it to explore new ideas related to large-scale additive manufacturing (AM), as well as other important topics such as multi-axis FFF and high-performance composite AM.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AM	Additive manufacturing
ABS	Acrylonitrile butadiene styrene
ASCII	American Standard Code for Information Interchange
BAAM	Big area additive manufacturing
BAAM-CI	Big area additive manufacturing system developed by ORNL and Cincinnati Inc.
CAD	Computer-aided design
CAM	Computer-aided manufacturing
CNC	Computer numerical control
FFF	Fused filament fabrication
HDPE	High-density polyethylene
I/O	Inputs/outputs
LCD	Liquid crystal display
LM2	Laboratory for Multiscale Mechanics
LSAM	Large-scale additive manufacturing
ORNL	Oak Ridge National Laboratory
PEEK	Polyether ether ketone
PEI	Polyetherimide
PLA	Polylactic acid
PSO	Position-synchronized output
PWM	Pulse-width modulation
ROS	Robot operating system
RTD	Resistance temperature detector
SSR	Solid state relay
TCP	Tool center point
TTL	Transistor-transistor logic
UART	Universal asynchronous receiver-transmitter

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Over the past decade, additive manufacturing (AM) has evolved from a rapid prototyping tool to a viable method of producing end-use parts [1]. Powder and resin-based AM technologies in particular have established themselves as versatile manufacturing processes in multiple industries, including aerospace and medicine. Fused filament fabrication (FFF), which deposits thermoplastics using extrusion, is the most widespread form of AM [2], [3], but it hasn't yet flourished as an industrial production method. To do so, FFF must overcome certain limitations.

Historically, FFF parts have suffered from poor surface finish, limited mechanical properties and small size [2]. Several groups have been working on addressing these weaknesses. To improve mechanical properties, many are developing fiber-reinforced composite materials and engineering-grade polymers specifically formulated for FFF [4]. Others are developing algorithms to improve printing toolpaths by leveraging the capabilities of multi-axis motion systems. These toolpaths make it possible to print stronger parts with improved surface quality. To produce larger parts, many groups transition to screw-based extrusion systems, substituting conventional filament feed stock for raw thermoplastic pellets. These systems enable higher deposition flowrates, which makes large scale printing more time efficient.

In the Laboratory for Multiscale Mechanics (LM2), researchers have studied the mechanical performance of FFF parts extensively, developing high-performance composite materials and specialized fabrication methods. The laboratory has also explored multi-material and multi-process printing. As part of this research, the LM2 used FFF and other AM processes to create multifunctional parts which couldn't be produced using conventional subtractive or formative fabrication methods. More recently, the laboratory has begun to investigate large-scale and multi-axis printing. To this end, the LM2 purchased two industrial motion systems: a large Aerotech gantry system with four independent z-stages and a Fanuc M-20iB/25 six-axis robotic arm. The Aerotech system can create large, multifunctional parts using multiple processes at once. Meanwhile, the Fanuc robot was purchased to explore even larger-scale printing using multi-axis toolpaths, enabling new fabrication strategies such as nonplanar printing. Custom printhead controllers produced by Montreal-based company Mëkanic allow users to print parts on both motion systems using different AM technologies.

Although these controllers have helped the team obtain important results, Mëkanic’s FFF-specific controllers were designed to drive small, high-temperature printheads exclusively (DyzEnd Pro hotend and DyzeXtruder Pro extruder). This project’s main objective was to design and fabricate a custom printhead controller adapted to large-scale printing on industrial motion systems. To future-proof the system, we prioritized controller modularity and flexibility with respect to toolheads: the goal was for a single controller to be capable of controlling a wide variety of toolheads on different motion systems. More specifically, we focused on creating a system capable of driving high-flowrate printheads, namely the Typhoon and Pulsar extruders by Dyze Design.

This thesis describes the steps we took to develop the printhead controller and use it to produce large-scale parts on the LM2 motion systems. The thesis is structured as follows:

- Chapter 2 overviews additive manufacturing in general and presents the state of the art in large-scale and multi-axis AM.
- Chapter 3 presents the project’s main objectives and provides background information on different topics we explored
- Chapter 4 describes the custom methodology we adopted to implement FFF on industrial motion systems.
- Chapter 5 overviews the printhead controller’s design, fabrication, and implementation process on both LM2 motion systems.
- Chapter 6 presents the large-scale printing results we obtained to demonstrate the system’s capabilities.
- Chapter 7 discusses the strengths and weaknesses of the system we developed based on the results obtained in Chapter 6

CHAPTER 2 LITTERATURE REVIEW

2.1 Additive manufacturing: general overview

Additive manufacturing (AM) – commonly referred to as 3D printing – is a fabrication process which creates geometries from 3D model data via successive addition of material, usually layer by layer [5]. This contrasts with more conventional subtractive manufacturing processes like milling or turning, which produce parts by selectively removing material from an initial blank. AM first emerged commercially in 1987, when 3D Systems released the SLA-1, which used stereolithography to build parts by focusing a laser on a UV-sensitive resin and solidifying it one layer at a time [6]. Since then, AM has expanded significantly: over the years, various processes have emerged, capable of printing a wide range of materials, including metals, polymers, ceramics, and composites. Originally relegated to prototyping non-functional parts, AM has become a versatile fabrication method "poised to reshape manufacturing" [1]. Apart from its rapid prototyping capabilities, AM offers several advantages over conventional subtractive and formative manufacturing methods [2], [7]–[10]:

1. *Production of complex geometries.* Because AM processes produce parts by automatically adding material where specified by 3D model data, additional complexity is typically achievable at no extra cost. This includes complex internal features which couldn't be created using conventional fabrication methods.

2. *Mass customization.* Unlike manufacturing methods like injection molding, which produce identical parts in high volumes according to a reference template, 3D-printed parts can easily incorporate individual modifications, such as orthotics adapted to specific patient geometry.

3. *Material compatibility.* AM can produce parts out of materials that are difficult to shape using conventional manufacturing processes, such as titanium or tungsten, which are difficult to machine conventionally due to their hardness.

4. *Part consolidation.* AM enables designers to combine multiple parts into one, simplifying assemblies and reducing part count.

5. *Local property control.* 3D-printed parts can locally feature specific physical properties adapted to their final use. For example, adding surface porosity to biomedical implants can facilitate ossointegration. Varying material composition or AM processes throughout a part can produce specific mechanical, chemical or electrical properties, creating multi-functional parts.
6. *Reduced waste.* Unlike subtractive processes, which must remove material from initial stock to produce final parts, AM can generate complex parts with minimal waste, depositing material only where required.

ISO/ASTM 52900:2021 divides AM technologies into seven broad categories [5], as presented in Figure 2-1 [7].

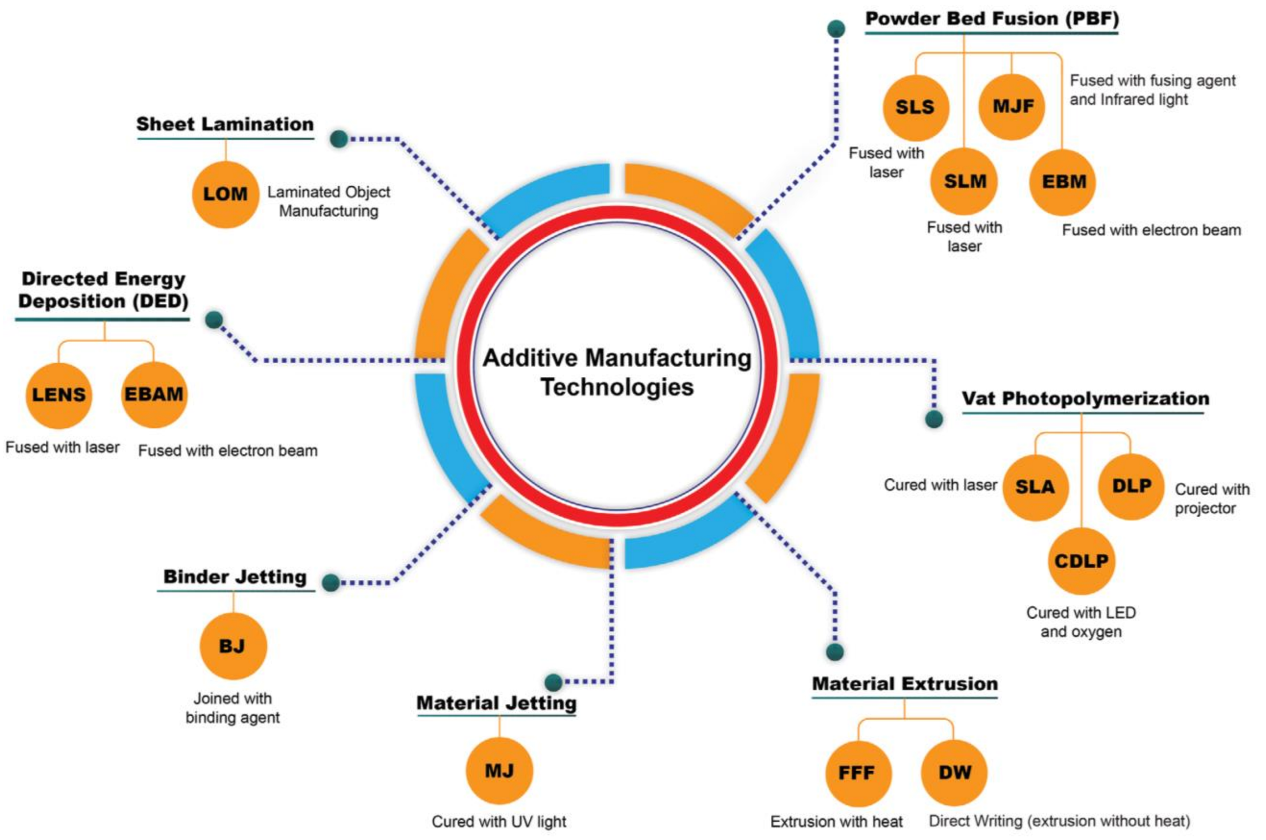


Figure 2-1: Seven main additive manufacturing process categories as classified by ISO/ASTM 52900:2021 [7]

In this review, we focus on material extrusion processes, which are the most widespread AM technology [2], [3] and the core of the LM2’s research on AM. In material extrusion, a material in semi-liquid or paste form is extruded through a nozzle or orifice and generally deposited layer by layer to form a final part. There exist two main material extrusion processes which are classified based on whether heat is involved in extrusion or not: fused filament fabrication (FFF) uses heat to melt a thermoplastic feedstock and extrude it, while direct writing (DW) processes extrude various types of paste-like materials without heating them. Fused filament fabrication, also known by its trademarked acronym FDM (for fused deposition modeling), is by far the most popular form of material extrusion and the subject of significant research and industrial interest in the past decade.

2.1.1 Fused Filament Fabrication (FFF)

Fused filament fabrication first appeared commercially in the early 1990s after Stratasys – a major manufacturer of AM printers and materials – filed a patent for the technology in 1989 [11] and trademarked the FDM term in 1991 [6], [12]. When this key patent expired in 2009, this generated a “wave of hype around 3D printing” [13] marked by the birth of the open-source RepRap 3D printer movement and many new companies eagerly joining the field. Numerous companies have since commercialized FFF printers with distinct features targeted at various industries, but these generally operate on the same basic principle shown in Figure 2-2.

In FFF, a thermoplastic filament is routed to a mobile extrusion head, often referred to as a printhead. In the printhead, a set of gears or a similar mechanism drive the filament towards a heated region (commonly referred to as the hot end, labelled *liquefier* in Figure 2-2), where it is melted and extruded through a nozzle. By moving the printhead in space in sync with the printhead drive gear speed, the nozzle deposits filaments which rapidly solidify. These filaments generally join to form solid layers, each of which represents a cross-section of the final part. The printhead builds these layers one on top of another, gradually producing a final part. As shown in Figure 2-2, certain printers possess multiple nozzles to extrude different materials: one common arrangement includes a main printing material for the final part geometry, and a sacrificial support material to enable more complex geometries.

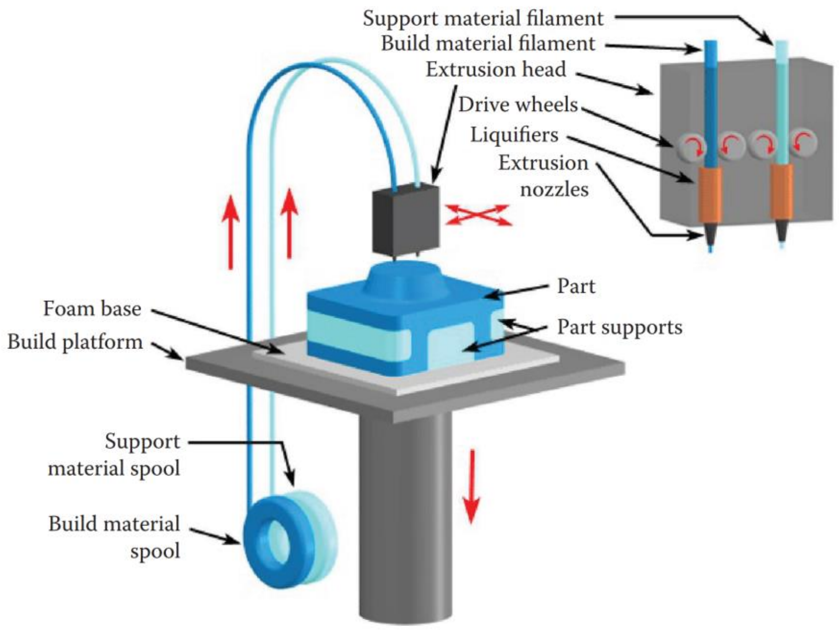


Figure 2-2: Schematic of the FFF process [14]

This way of producing parts has several advantages. On top of the general AM advantages presented previously, FFF is typically low-cost [3] and compatible with a wide variety of materials [1], [14]. It's also particularly well-suited to printing multiple materials at a time, producing multi-functional parts which exhibit specific location-dependent properties [3]. FFF also possesses several drawbacks: compared to other AM processes, extrusion-based AM typically creates lower resolution parts, struggling with fine features [2] and producing rougher surface finishes, namely due to the stair-stepping effect of stacked layers [14]. FFF parts are also limited by the mechanical properties of available thermoplastics [2], whose strength and stiffness are inferior to those of metals used in other AM technologies [4]. This is compounded by several process limitations which can weaken FFF-produced parts even more, such as gaps between deposited filaments leading to weaker interlayer adhesion and contributing to FFF part anisotropy [15]. FFF is a complex manufacturing method with many process variables affecting final part appearance and mechanical properties; these can lead to defects and inconsistent part quality [9]. Finally, FFF printers typically possess limited build volumes and slow production speeds [4], [10]. These limitations have historically restricted their use in many industries, relegating them to rapid prototyping and very specific applications.

Over the past decade, many industrial and academic groups have worked towards addressing these limitations. On the materials side, researchers and companies alike have explored the use of composites to improve the mechanical properties of FFF parts; carbon fiber-reinforced polymers have become particularly prominent in the field due to their promise of high strength and stiffness while reducing part weight [4]. The LM2 has studied this topic extensively over the past few years as part of the Safran Industrial Research Chair on Additive Manufacturing of Organic Matrix Composites (AMOMC) [16]. On the production side, several groups have adapted the FFF process to rapidly produce large parts, as shown in the following sections. There are typically two main approaches to scaling FFF: using large, gantry-style printers similar to those seen at the desktop level; or using multi-axis industrial robots.

2.2 Large-scale additive manufacturing

Large-scale additive manufacturing as a term can encompass a wide variety of processes operating at completely different scales, with FFF parts the size of chairs and concrete buildings produced using AM falling under the same umbrella. In this section, we mainly address FFF and similar AM technologies without imposing restrictions on scale.

Oak Ridge National Laboratory (ORNL) in the United States has been heavily involved in much of the research on large scale additive manufacturing. In 2013, the research laboratory partnered with Lockheed Martin to develop a Big Area Additive Manufacturing (BAAM) machine [17] to produce parts “completely unbounded in size” [10]. To achieve this goal, they created a melt extrusion-based printhead which deposits high-performance engineering thermoplastic composites similarly to conventional FFF printheads [10], [17], but at a larger scale. Unlike FFF printheads, which use spooled pre-extruded filaments as a feedstock material, the BAAM system developed by Lockheed Martin and ORNL directly uses polymer pellets, powders, fiber reinforcements and specialty additives, which allows it to produce higher performance composites at a lower cost [10]. This deposition head is mounted to a multi-axis robot arm, which can either be stationary or mounted to a large, three-axis gantry system to produce even larger parts [10]. Alternatively, the same deposition head can be directly mounted to a conventional three-axis system [10]. Figure 2-3 illustrates the BAAM system developed by Lockheed Martin and the ORNL. The team’s long-term vision is to use multiple robotic deposition systems operating synchronously to produce large components such as wings of unmanned aircraft [10], [17].

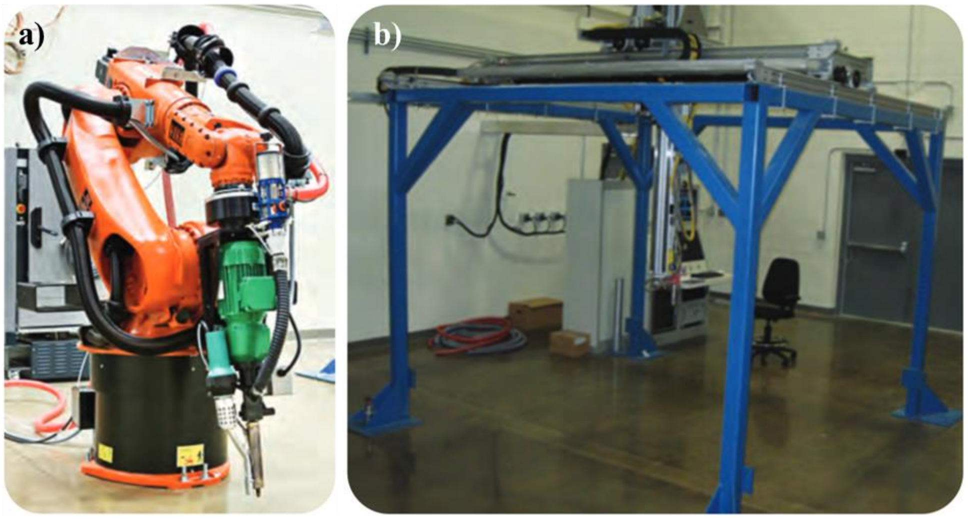


Figure 2-3: a) Robotically controlled BAAM printhead developed by ORNL and Lockheed Martin [10] b) ORNL three-axis AM deposition system compatible with BAAM printheads [10]

In 2014, ORNL partnered with Cincinnati Inc. to create another BAAM system [17], often referred to as BAAM-CI. Like the Lockheed Martin system, it uses thermoplastic pellets and fiber reinforcements as a feedstock material. These are fed into a single screw extruder which melts and deposits them to produce parts. This type of pellet-extrusion system is similar to FFF but achieves much higher flowrates: while desktop-sized AM systems reach flowrates of 16 to 82 cubic centimeters per hour, flowrates on the BAAM extruder can exceed 16 000cm^3/h [18], producing larger parts much faster. The BAAM extruder is installed on a commercial three-axis gantry system by Cincinnati Inc. which allows it to reach speeds of over 5m/s with a positional accuracy of 0.05mm [18]. The BAAM-CI system's print volume is 2.44m by 6.01m by 1.83m [19]. Figure 2-4a shows the system printing a vehicle frame.

Different groups have used the BAAM-CI system for a variety of applications. One common theme which appears in several publications is large-scale mold production [18]–[23]. In 2017, Post et al. fabricated a 13-meter-long two-part wind turbine blade mold using 16 individual BAAM printed sections [18]. These sections were then covered in fiberglass and machined to their target dimensions using a different system. Figure 2-4b shows one such section (approximately 1.8m tall) after having been printed vertically on the BAAM-CI system, while Figure 2-4c shows the two final mold parts ready for use. In 2019, the same team used the BAAM-CI system to produce a 10.36m long boat hull mold in six sections using a similar process [19], while a different group used it to produce molds for hydroelectric components [22].

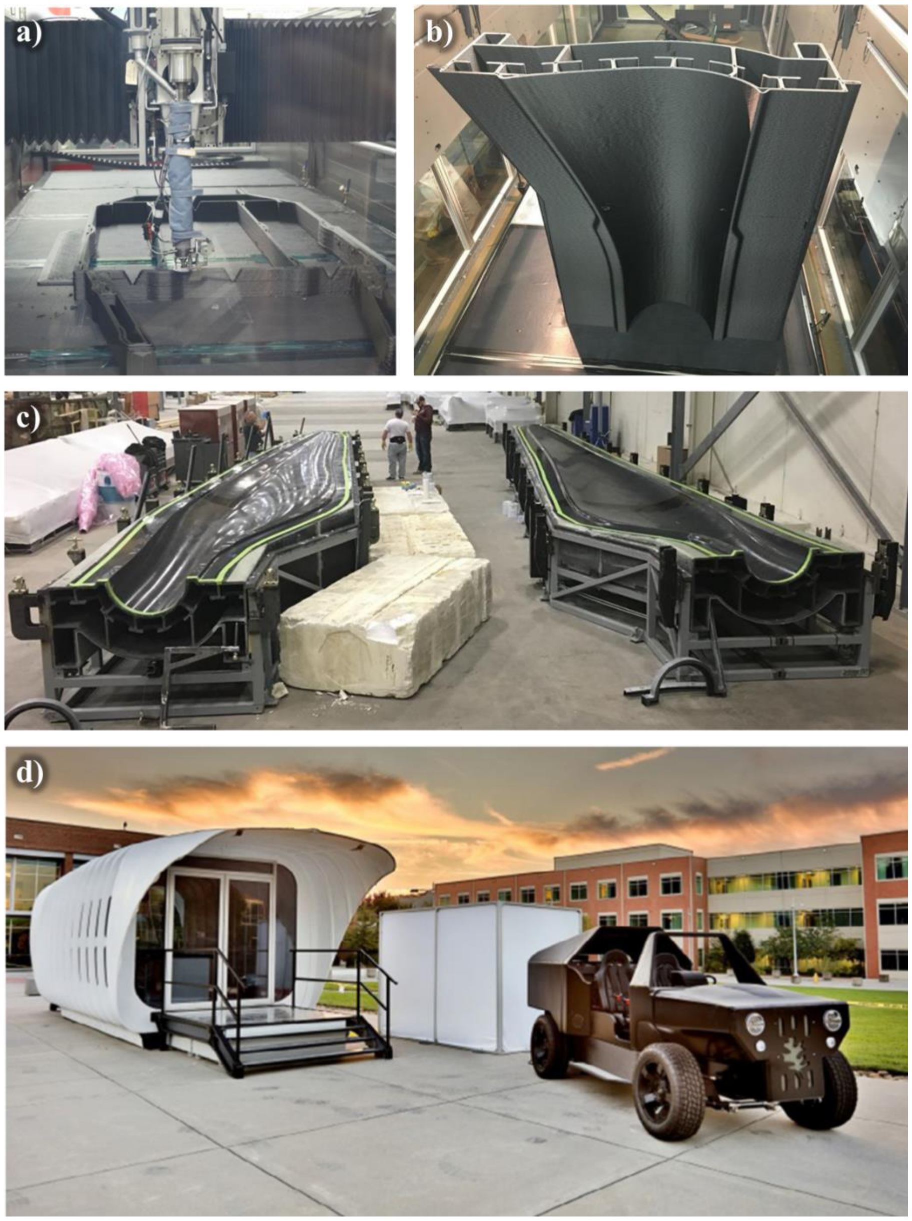


Figure 2-4: a) BAAM-CI system printing a vehicle frame [24] b) 1.8-meter-tall wind turbine blade mold segment printed on the BAAM-CI system [18] c) Completed two-part wind turbine mold after fiberglass application and machining [18] d) Single-room building and range-extended electric vehicle printed on the BAAM-CI system [24]

The BAAM-CI system has also been used to print vehicles and small buildings. In 2014, ORNL partnered with Local Motors and Cincinnati Inc. to print the first Strati electric car in 44 hours at the International Manufacturing Technology Show (IMTS) in Chicago [25]. The car was then machined in less than 12 hours and assembled in less than 24 hours. Reaching this point required several iterations, with the team having to increase the extruder's flowrate and address thermal issues causing delamination between layers before being able to successfully print the car in one take. In 2015, Talagani et al. presented a numerical simulation methodology to analyse distortion and built-up residual stresses in AM parts, which they used to simulate the printed Strati car's manufacturing process and identify problematic areas [26]. In 2017, Chambon et al. presented the development of a range-extended electric vehicle and a single-room building printed with the BAAM-CI system, as shown in Figure 2-4d [24]. The entire vehicle frame and body were printed, while the building incorporated vacuum-insulated panels in a BAAM-printed shell [24].

Several groups have published innovative research completed using the BAAM-CI system:

- In 2016, Duty et al. analyzed the mechanical behavior of BAAM materials. As part of their investigation, they developed an air-cooled tamping attachment for the BAAM printhead to compact deposited beads as they exit the nozzle, creating a more uniform surface and significantly increasing the strength and stiffness of printed parts [27]. Figure 2-5a and b show the tamping system they developed, while Figure 2-5c shows the effect on tensile strength.

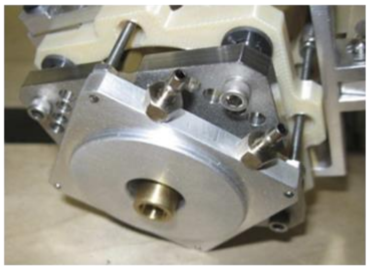
- In 2017, Chesser et al. created a selectable nozzle design that allows users to dynamically vary print resolution with a single BAAM extruder, as shown in Figure 2-6a. They used the smaller nozzle to print the outer surface of different parts while using the larger nozzle for most of the print, improving surface finish without dramatically increasing print time [28].

- In 2018, Roschli et al. devised a method to avoid the types of delamination which appeared on the Strati electric car printed on the BAAM-CI system: this method involves leaving holes in the printed layers, and then filling them a few layers later, effectively creating pins in the Z-axis to improve interlayer properties [29]. This method is illustrated in Figure 2-6b.

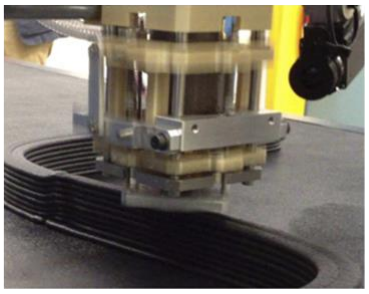
- In 2019, Atkins et al. developed a co-extrusion tool which allowed them to embed wires in parts produced using BAAM [23]. Billah et al. used this system in 2021 to embed heated elements in large-scale parts, printing heated molds to improved part curing without the additional manual work usually involved in routing such elements [20]. Figure 2-6d shows the coextrusion tool installed on the BAAM-CI system, while Figure 2-6e illustrates how it works.

- In 2020, Smith et al. fabricated a dual-material switching system, allowing them to print two materials with a single BAAM pellet extruder [30] as shown in Figure 2-6c. In 2021, Brackett et al. characterized the transitions when switching materials with this dual-hopper system, analyzing their impact on the deposited material's composition [31].

a)



b)



c)

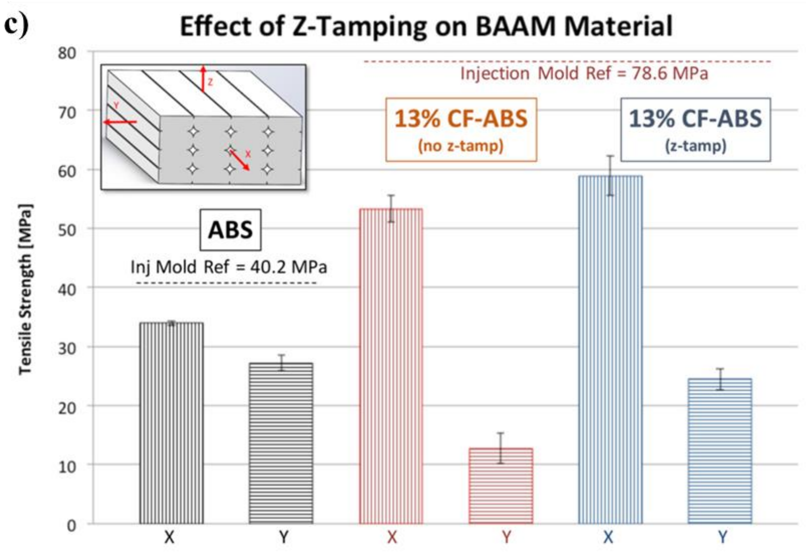
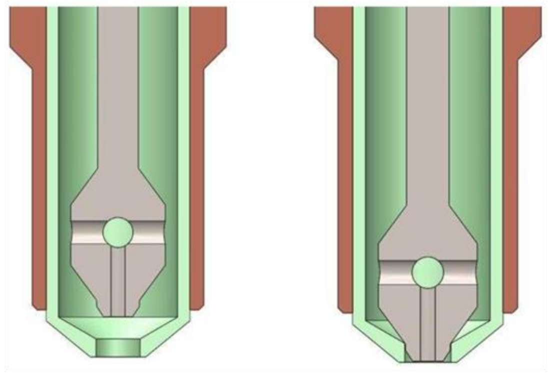
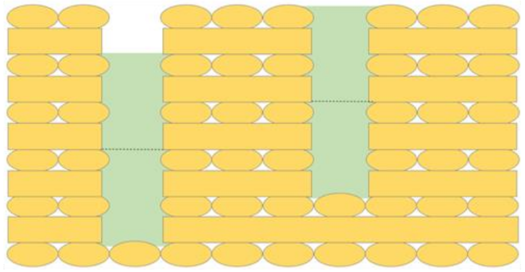


Figure 2-5: a) Z-tamping system created for the BAAM-CI system [27] b) Tamping system in use [27] c) Effect of z-tamping on tensile strength of printed samples [27]

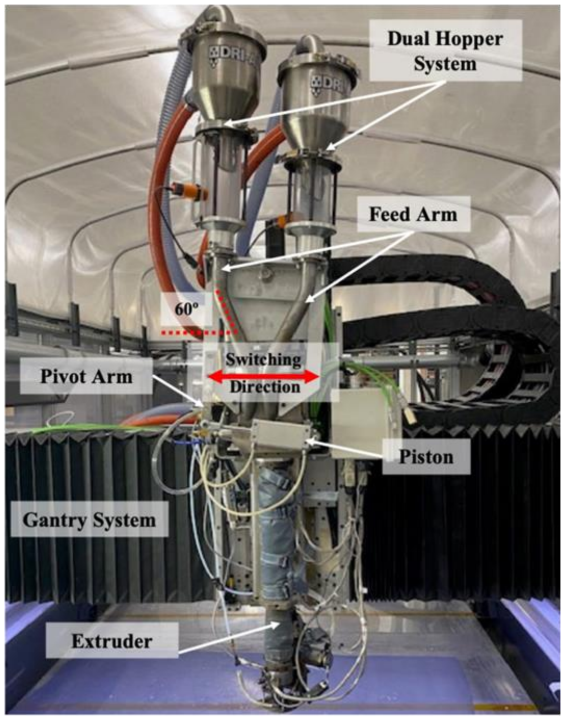
a)



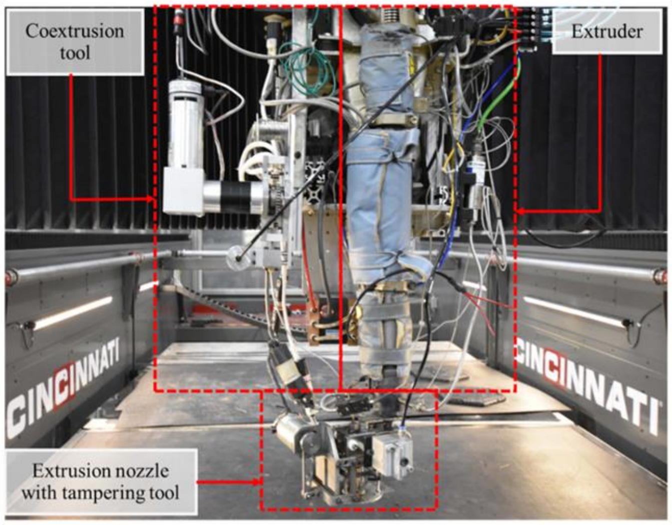
b)



c)



d)



e)

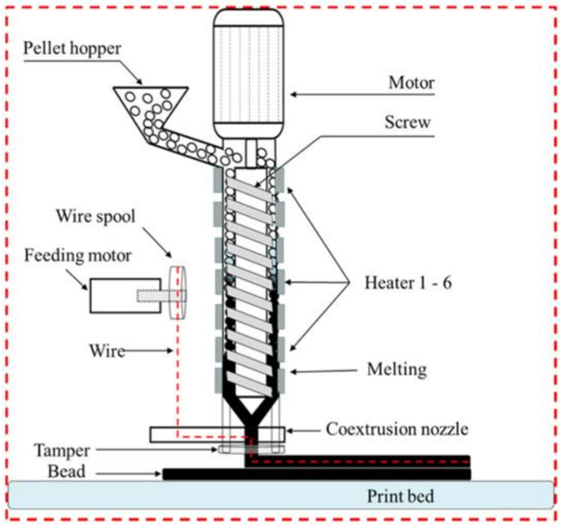


Figure 2-6: a) Dual-diameter selectable nozzle system [28] b) Z-pinning method to increase interlaminar strength in BAAM-printed parts [29] c) Dual-material switching apparatus installed on the BAAM-CI system [30] d) Wire coextrusion tool installed on the BAAM-CI system [20] e) Schematic of the wire coextrusion tool [20]

While the BAAM-CI system appears in much of the published research on large scale additive manufacturing, it’s far from the only existing large-scale AM system. Thermwood Corporation produces several Large-Scale Additive Manufacturing (LSAM) printers in different sizes, with build volumes ranging from 1.5 x 1.5 x 1.2 m^3 to 4.6 x 12.2 x 1.5 m^3 and beyond. Many of these printers also feature milling heads, making it easier to print and post-process large parts on the same system [32]. Figure 2-7 show two such LSAM systems printing large scale parts. In 2020, Fathizadan et al. used thermal cameras to monitor the real-time temperature of parts being produced on an LSAM machine [33]. They used the captured images to produce a thermal data extraction and layer control time framework, allowing them to predict upcoming layer temperatures and optimize layer print times. Apart from this 2020 publication, there has been little published research produced with an LSAM system so far. This is likely to change in upcoming years: in 2021, Purdue University’s Composites Manufacturing Simulation Center (CMSC) and Thermwood Corporation established the Thermwood Research Center to perform industry-funded research on large-scale thermoplastic composite AM [34].

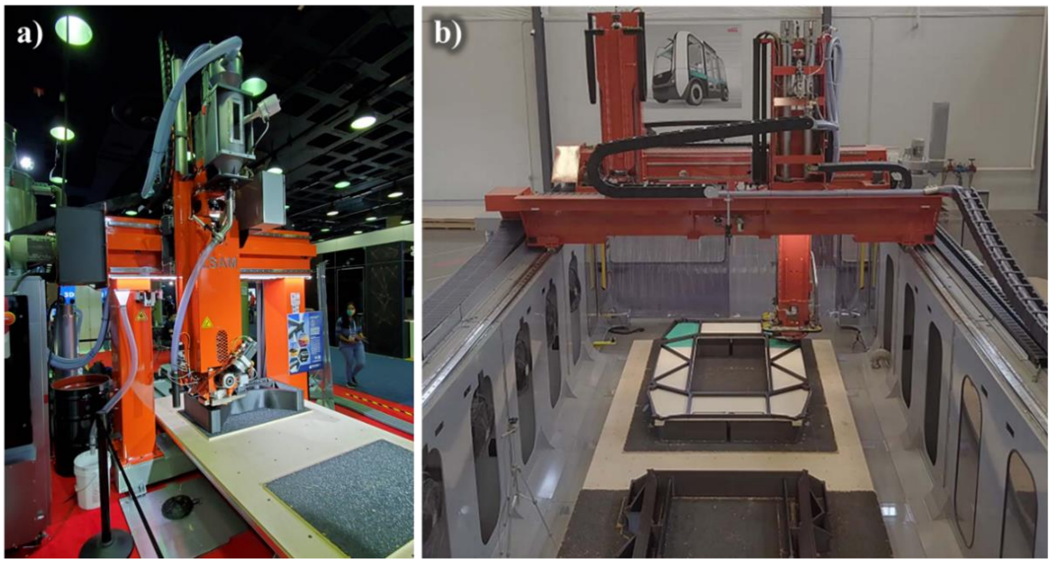


Figure 2-7: a) LSAM 510 AP additive-only system printing a mold at the 2021 RAPID + TCT trade show in Chicago b) Larger LSAM system printing a lower car chassis [33]

A few other large-scale systems feature in research publications:

- In 2018, Nieto et al. developed a large format, cartesian AM system for the naval industry. This printer, called the S-Discovery, has a printing volume of over 3 cubic meters and uses a screw-based pellet extruder for deposition [35]. Sánchez et al. used this printer in 2020 to help develop carbon fiber acrylonitrile styrene acrylate composites for large-scale printing [36].

- In 2020, Nycz et al. investigated the effect of using infrared heating lamps to locally increase the substrate temperature of large-scale AM parts right before depositing new material. This allowed them to improve the interlayer mechanical properties of samples they printed on an experimental Blue Gantry system, which possesses a build volume of 2m by 2m by 1.3m [37]. This study resembles a 2017 publication by Kishore et al., who also implemented a local infrared heating solution on a BAAM-CI system and observed similar results [38].
- In 2021, Pappas et al. installed a custom single-screw extruder on a Galaxy G motion system by Automated Precision Inc., which they used to produce carbon-fiber reinforced composites, using different extrusion methods to vary the fiber length in the resulting parts [39].

Outside of the academic field, numerous companies produce equipment for large scale additive manufacturing. Aside from specialized multi-axis systems, which we address in the next section, notable systems include:

- Wide and High Additive Manufacturing (WHAM) system by Ingersoll, developed through a cooperative agreement with ORNL starting in 2016 and officially announced in 2018. The WHAM system has a build volume of 6.1 x 18.3 x 3.0 m^3. [40]
- Atlas 3D printers by Titan Robotics. These possess 1.27 x 1.27 x 1.83 m^3 build volumes and feature filament extruders, pellet extruders, and milling toolheads. [41]
- Large-scale AM equipment by Massive Dimension, including pellet dryers, modular heated build plates, and pellet extruders outputting 4.5kg/h. [42]
- Typhoon filament extruder and Pulsar pellet extruder by Dyze Design. We address these printheads in Section 5.1.1.

2.3 Multi-axis additive manufacturing

In this section, we discuss AM on systems featuring more than three axes, which we refer to as multi-axis additive manufacturing. Although there are many interesting ways of implementing multi-axis AM – such as adding additional axes to a conventional 3D printer [43]–[46] or developing a custom positioning system based on cables [47] or linkages [48] – the overwhelming majority of research publications rely on robotic arms. Most groups who use this system mount an extrusion printhead to the robotic arm, as shown in Figure 2-3a; the robotic arm moves the printhead in space, following pre-established toolpaths, while the latter deposits material like in the conventional FFF process.

This method is well suited to large scale additive manufacturing: as we saw in the previous section, while conventional AM systems must grow significantly to produce larger parts – with maximum part volume never exceeding the size of the printer itself – robotic arms have a much greater workspace relative to their size [49], [50]. Multiple robots can also collaborate within the same space to produce even larger parts: in 2018, Zhang et al. used two mobile six-axis robots to concurrently print a large concrete structure, as shown in Figure 2-8a. To do so, the team mounted the robots to holonomic platforms and fitted them with stereo camera systems. From a pre-established home position, each robot used onboard sensors and its camera system to navigate to a fixed location. While stationary, it then began printing its portion of the structure, before returning to its home position [51]. In 2019, Tiryaki, Zhang et al. expanded on this work by implementing "printing-while-moving," allowing a single mobile robot to print large, single-piece structures [52]. On a smaller scale, Shen et al. used four robotic arms in 2019 to collaboratively print a single part, reducing print time from 10h44m to 2h53m [53]. Figure 2-8b shows a schematic of the process.

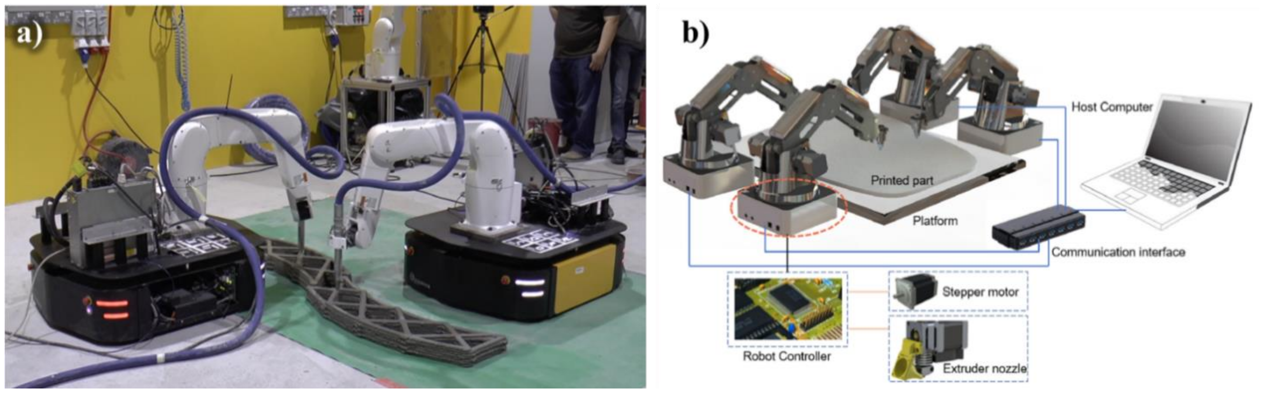


Figure 2-8: a) Two mobile robots concurrently printing a large concrete structure [51]
b) Schematic of four robotics collaboratively printing a part [53]

While robotic arms shine at large scale AM, their main strength lies in their ability to follow complex toolpaths, offering more manufacturing flexibility than traditional AM. Because conventional FFF systems possess only three axes, they are typically limited to creating parts by building up layers one on top of another, sometimes referred to as 2.5D printing. This stacked-layer process leads to several of the weaknesses we mentioned in Section 2.1.1, such as surface stair-stepping and part anisotropy. It also requires the use of support structures to build steep overhanging features, increasing waste and printing time. On the other hand, robotic arms usually feature six or more axes; these extra degrees of freedom mean they can orient printheads in ways that allow them to overcome some of these limitations. This has prompted several groups to explore their use in non-planar AM. [54]–[73]

One non-planar AM method which has received much attention in the robotic arm-assisted FFF community is sometimes referred to as “conformal printing” [58]. This method involves printing the sloping regions of parts using continuous, non-planar toolpaths instead of conventional stacked layers. This produces significantly better surface finish, removing the stair-stepping effect commonly observed on FFF parts [58]. It can also improve mechanical properties and reduce print times [58]. Although this method was initially developed by Allen and Trask in 2015 for use on a conventional FFF printer [74], multi-axis systems can push it even further: by changing the printhead’s orientation appropriately, these systems can print highly sloped parts without the toolhead colliding with printed geometry.

Several groups have experimented with conformal printing to produce parts big and small:

- In 2015 and 2016, Zhang et al. at the ABB US Corporate Research Center and Arevo Labs Inc. implemented conformal printing on a robotic arm, using ABB's RobotStudio software to generate non-planar toolpaths. [71], [72]
- Between 2017 and 2022, several groups at the Center for Advanced Manufacturing at University of Southern California investigated different forms of conformal printing using robotic arms from ABB and Yaskawa. They developed slicing, path planning, and collision detection algorithms which they then used to produce numerous parts. Figure 2-9a shows an ABB IRB-120 robotic-arm conformally printing a thin-walled sample, which Alsharhan et al. submitted to compression tests in 2017. Compared to an equivalent part produced using conventional FFF, the non-planar toolpaths led to higher stiffness and peak loads, but more sudden structural failure. Figure 2-9b shows another part created using two ABB robotic arms printing collaboratively. The robots were fitted with two different nozzle sizes to reduce printing time while optimizing final part quality. [57], [58], [62], [64], [73]
- In 2018, Zhao et al. used a KUKA robotic arm to conformally print a large, complex part on a cylindrical printing surface, as shown in Figure 2-9c and d. To generate the toolpaths, they flattened the original cylindrical 3D model, sliced the flat version into planar toolpaths using a conventional slicing software (see Section 3.2.1.2 for more details about slicing), then applied the reverse transformation to the planar toolpaths to obtain cylindrical ones. [66]
- In 2021, Yao et al. used a UR3 collaborative robot to produce several demonstrative parts, including a conformal print on a conventionally printed planar support structure, as shown in Figure 2-9e. They used a similar method as Zhao et al above to generate nonplanar toolpaths. [54]

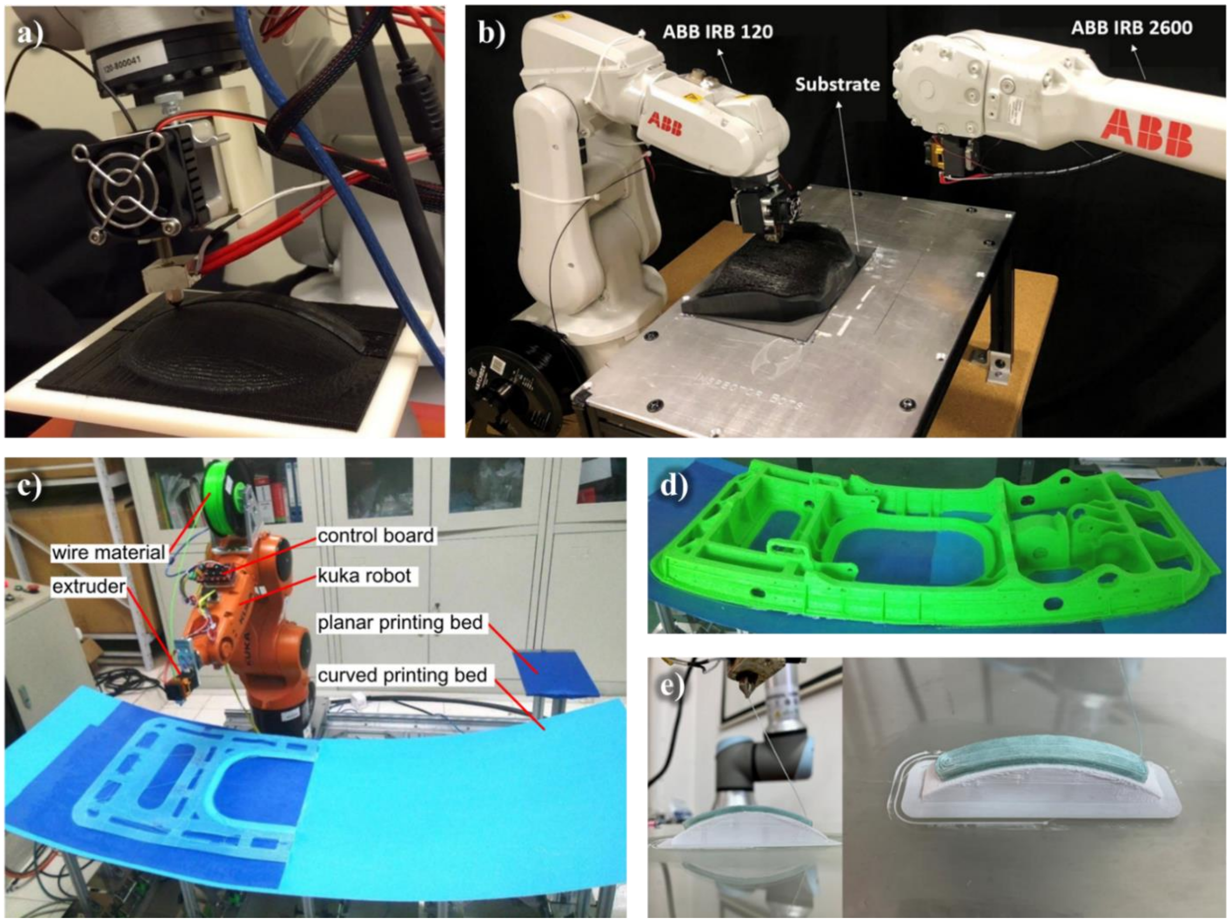


Figure 2-9: a) Conformally printing a thin-walled specimen [62] b) Using two robots with different-size nozzles to conformally print non-planar parts with varying resolutions [57] c) Conformally printing a complex part on a cylindrical surface [66] d) Resulting part obtained after c [66] e) Conformally printing a sample using a collaborative robot [54]

Outside of conformal printing, a few research groups have explored other forms of nonplanar printing using robotic arms. In 2016, Ishak et al. developed a robotic arm FFF system based on a Motoman SV3X robot which they then used in 2017 to print 3D lattice structures, as shown in Figure 2-10a [68], [69], [75]. In 2017, Wu et al. developed a slicing algorithm to produce parts using FFF without support material; they then used a UR3 collaborative robot to create parts according to their algorithm, as shown in Figure 2-10b [67]. Finally, Felbrich et al. developed an entire robotic arm-based AM platform inspired by the shell formation of land snails in 2018. They created a custom printhead based on an elongated die connected to a single screw extruder, which they then mounted on a six-axis robotic arm. They used this system to deposit large strips of HDPE along freeform trajectories, as schematized in Figure 2-10c [56].

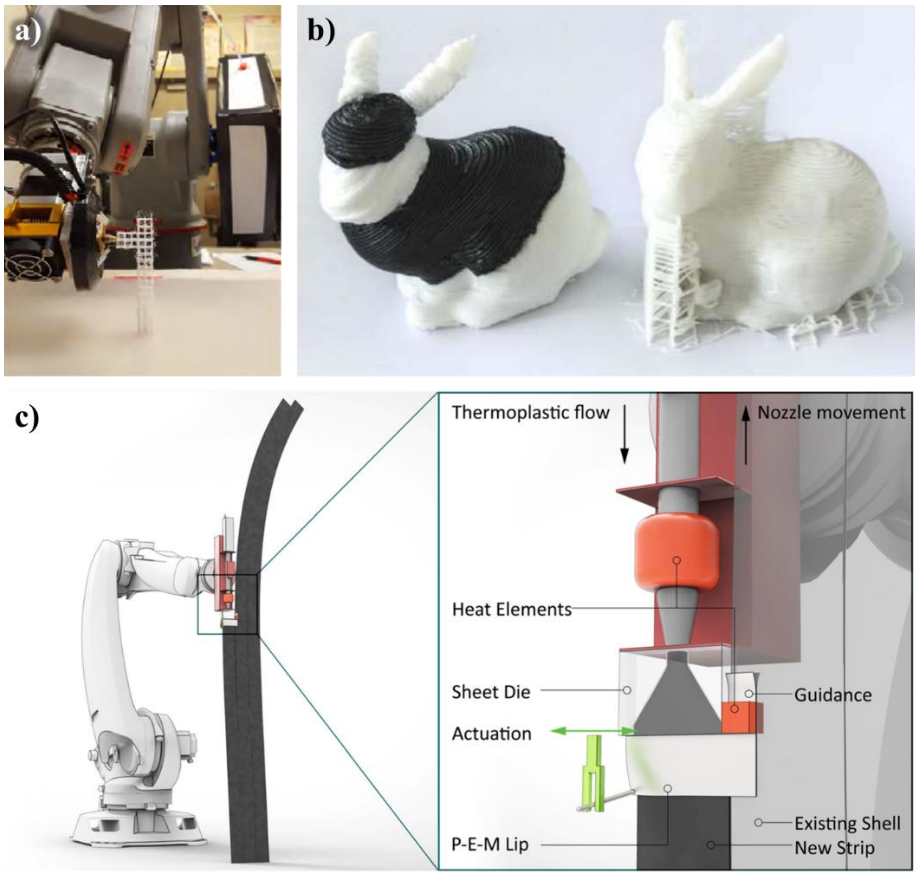


Figure 2-10: a) Robotic arm-based FFF printing of 3D latices [68] b) Comparison between a part printed using Wu's non-planar algorithm (left) and using conventional slicing (right) [67] c) Schematic of a snail shell-inspired method of freeform AM [56]

While most groups investigating robotic arm-assisted AM mount a printhead to a robotic arm, depositing material onto a fixed build plate, certain groups prefer to attach the build surface to the robotic arm instead [59], [63], [65], [67], [70], [76]. Although this can be useful in certain scenarios, it generally leads to a significant reduction in build envelope, making it less suitable for large-scale AM.

Like some of the large-scale AM systems covered in the previous section, certain research groups have implemented multiple processes on their robotic arm-based AM systems, combining additive, formative and subtractive methods [55], [60]. In 2013, Keating et al. developed the most diverse multi-functional robotic platform to date, producing parts using an FFF printhead, a single-screw extruder, a spray foam deposition toolhead, a sculpting toolhead, and a milling toolhead.

In industry, very few companies produce AM-specific multi-axis equipment available for purchase:

- CEAD is an AM-specific equipment supplier, providing both gantry and robot-based large-scale AM systems. [77]
- On the software side, Ai Build produces toolpath generation, process control and monitoring software targeted at non-planar printing using multi-axis systems. They partner with robotic manufacturers like KUKA, ABB and Stäubli and large-scale AM equipment suppliers like Dyze Design, but do not produce hardware themselves. [78]
- ABB [79], Yaskawa [80], and KUKA [81] are the only well-known six-axis robot manufacturers we found with web pages advertising AM capabilities, but none of these systems are AM-ready as provided. ABB's Robot Studio includes a 3D printing package to print parts directly from CAD [79].

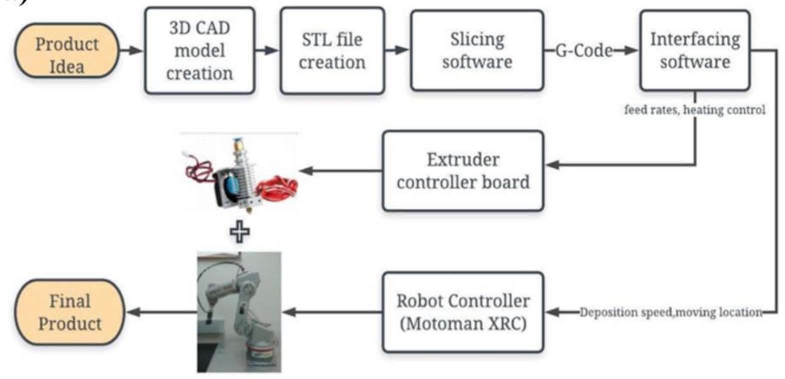
Because of this, most of the research groups overviewed in this section added AM capabilities to existing industrial multi-axis motion systems. We discuss this in more detail in the next section.

2.4 Adding additive manufacturing capabilities to multi-axis industrial motion systems

To produce parts using multi-axis AM, most research groups mount commercial or custom-made printheads to a robotic arm, as we saw in the previous section. Because robotic arms are rarely capable of interfacing with printheads out of the box, these groups must then add on the required electronics to control the printheads and create custom AM workflows adapted to their hardware setup. This section overviews some of the main methods which appear in research publications.

To control the printheads, most publications feature an Arduino-based solution [54], [55], [58], [59], [62], [66]–[69], [75], [82]–[86]. Nearly all of these use an Arduino MEGA 2560 board paired with a RepRap Arduino Mega Pololu Shield (RAMPS) running Marlin or Repetier firmware. This electronics arrangement is typically used in FFF printers, as it possesses the various inputs and outputs required to drive stepper motors, heaters, and other common components found in such machines. Many groups configure these electronics to control a printhead exclusively, while the robotic arm controller independently drives the overall system's motion and position in space. Producing parts using these two independent systems conjunctly requires a custom workflow, with most groups adopting a method like that shown in Figure 2-11a. In this workflow, the g-code file outputted by a slicing software is separated into two sets of instructions: extrusion feed rates and heater instructions are sent to the extrusion controller, while motion instructions are sent to the robotic arm [87]. Amongst several others, the parts shown in Figure 2-9a, d, and e, and Figure 2-10a and b in the previous section were created using an Arduino+RAMPS-based extrusion control system and this kind of workflow.

a)


Product Idea
3D CAD model creation
STL file creation
Slicing software
G-Code
Interfacing software
feed rates, heating control
Extruder controller board
Robot Controller (Motoman XRC)
Deposition speed,moving location
Final Product

b)

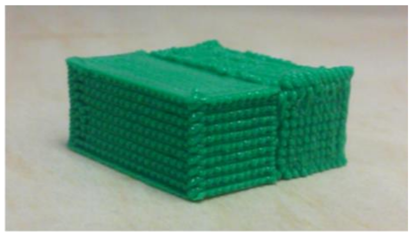


Figure 2-11: a) Robotic-arm FFF part-production workflow used by Ishak et al. [75] as synthesized by Urhal et al. [87] b) Multi-planar part printed by Ishak et al. featuring over-extrusion during direction changes [75]

Although several groups have used this method to successfully print parts, it suffers from one main weakness: extrusion and motion commands are independent. This can lead to poor print quality under certain circumstances, such as when the robot accelerates and decelerates. Although this didn't cause major issues for the above groups – who used small nozzle sizes and printed at mostly constant speeds [54], [59], [66], [85] – it did cause minor defects. Figure 2-11b shows a multi-planar part printed by Ishak et al. [75] which features over-extrusion in the corners and in other areas involving sudden direction changes.

To synchronize printhead extrusion with robot motion, certain groups have chosen a different extrusion control method which relies on analog signals sent from the robotic arm controller to the extrusion controller [64], [72], [73], [84], [88]. In this method, the robotic arm controller generally outputs an analog signal proportional to its real-time tool center point (TCP) speed. It sends this analog signal to the extrusion controller, which adjusts the material deposition rate accordingly. Zhang et al. first used this method to produce non-planar parts using an ABB robotic arm in 2015, as mentioned in the previous section. They used ABB's DispenseWare package to send the analog TCP speed signal to a PG20 pulse generator, which sent pulses to a stepper driver connected to the extruder motor. As the analog signal varied, the pulse frequency followed, thus maintaining proper synchronization between printhead deposition and robot motion [72].

Badarinath and Prabhu produced the most exhaustive implementation and analysis of this extrusion control method. In their 2021 publication, they created a custom extrusion controller capable of driving an E3D Titan Aero extruder driven by an integrated stepper motor with encoder feedback, as well as a heated print bed. They connected the printhead and extrusion controller to an ABB IRB 140 industrial robot, establishing communication between all systems according to the architecture shown in Figure 2-12a. The two researchers then characterized the analog TCP speed signal produced by the robot controller as shown in Figure 2-12b. They studied its relationship with the robot's actual TCP velocity and intentionally simulated signal delays to observe their effects on printed parts. As part of this investigation, the group produced multiple benchmark parts, measuring the dimensions of extruded filaments in straight segments, curves, and corners of varying angles (see Figure 2-12c). [88]

One final topic which deserves mentioning is Robot Operating System (ROS). ROS is an open-source meta operating system meant for programming robots [89]–[91]. It consists of packages, libraries, and other software tools meant for a variety of robotics applications, allowing developers to integrate various electronic components into a single system while using a common programming language. Packages exist for numerous industrial robots, such as robotic arms from ABB or KUKA. This prompted certain research groups to develop their FFF robotic arm software architecture in ROS. Kubalak et al. developed a custom extrusion controller in 2016 which they connected to an ABB robot using ROS; they used this system to produce various parts, including tensile test specimens which they printed in different orientations [61]. Wu et al. also used ROS for their FFF robotic arm application: they connected an Arduino-based MKS Gen board to an UR3 collaborative robot, which they used to produce parts according to a custom slicing algorithm. We discussed this in the previous section, with one such part shown in Figure 2-10b [67].

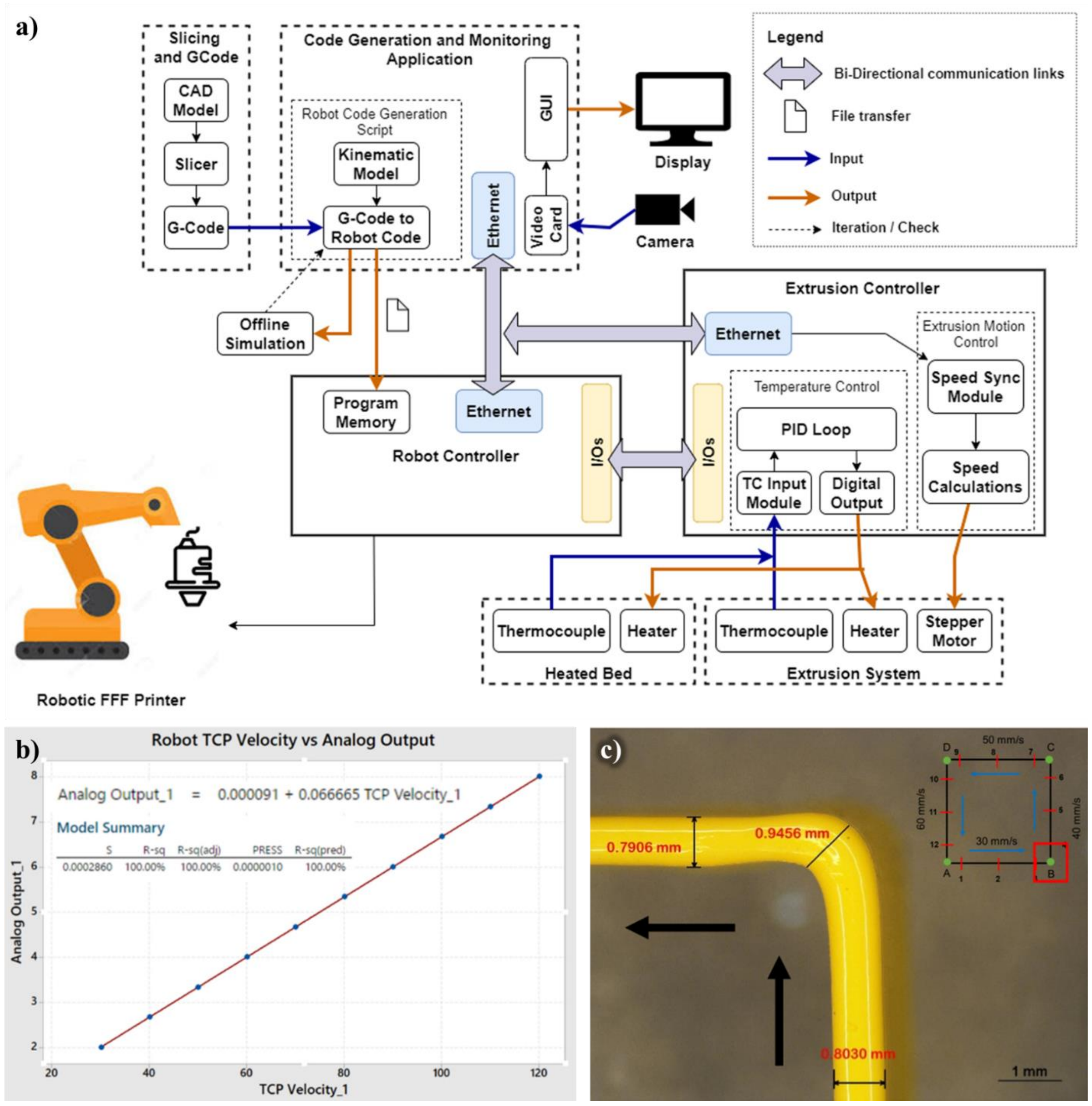


Figure 2-12: a) Control architecture implemented by Badarinath and Prabhu to implement FFF on an industrial robotic arm b) Comparison between the actual TCP velocity and the corresponding analog signal c) Benchmark test print to characterize analog signal delays on extruded filament dimensions [88]

CHAPTER 3 PROJECT DEFINITION

Current research identifies part scale as one of FFF's major weaknesses, preventing it from becoming an established industrial manufacturing method. In the previous chapter, we presented the work of several groups who have begun exploring alternate forms of FFF to address this weakness. Potential solutions often involve installing high flowrate printheads on industrial motion systems, which is of particular interest to the LM2 as the laboratory begins exploring large scale printing on the Aerotech and Fanuc systems. Although certain groups have developed custom controllers to drive specific printheads, we found no documented examples of printhead controllers capable of controlling various types of toolheads on multiple motion systems.

3.1 Main objectives

This project's main goal was thus to create a modular printhead controller adapted to large-scale FFF on industrial motion systems. More specifically, the controller needed to be capable of:

a. Controlling the Typhoon and Pulsar extruders from Dyze Design, as well as other FFF toolheads which may be of interest to LM2 researchers in the future.

b. Operating on both the Aerotech gantry system and the Fanuc six-axis robot.

To achieve this goal, we completed the following sub-objectives:

1. Developing a custom methodology for FFF printing on industrial motion systems.
2. Designing, fabricating, and integrating the universal printhead controller on LM2 motion systems.
3. Printing large-scale parts to demonstrate the system's capabilities.

The following chapters describe the work we completed to achieve each of these sub-objectives.

3.2 Background information

This section contains details which may help readers more thoroughly comprehend the work completed in future chapters. It includes background information on the conventional FFF part-production workflow, common printhead components, serial communication, and the commonly printed #3DBenchy benchmark part.

3.2.1 Conventional FFF part-production workflow

To develop a custom workflow to implement FFF on industrial motion systems, one must first understand the conventional FFF workflow used on most commercial printers. As illustrated in Figure 3-1, producing parts on a conventional FFF printer usually involves four main steps. The following sections describe each of these steps in more detail.

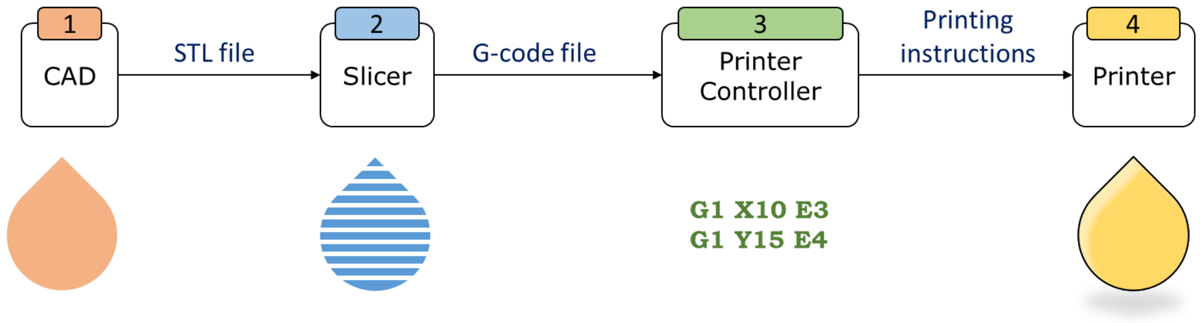


Figure 3-1: Conventional workflow for producing parts on a FFF printer

3.2.1.1 Step 1: Generating an STL file describing the part to print

The STL file format is widely used in additive manufacturing and contains triangulated surface data approximating the outer surface of the geometry to print. Most computer-aided design (CAD) software can export this file format at varying resolutions, allowing users to print their own designs. Alternatively, STL files for various designs can be shared and freely downloaded or purchased from a wide array of websites such as Thingiverse.com [92] or Prusaprinters.org [93].

3.2.1.2 Step 2: Converting the STL file into a G-code file

This step happens in a *slicer*, a 3D-printing software that divides the part geometry into stacked, printable layers according to user-defined settings. The slicer allows users to set various printing parameters such as nozzle temperatures and printing speeds and optimize toolpaths according to the part’s geometry, material, and application. Users can add multiple parts to the slicer at once for them to be fabricated during the same printing process. The output G-code file contains the set of instructions required to print the parts. These instructions follow a standard format and fall into three main categories:

1. G-commands for all motion-related instructions. When the slicer divides one or many parts into layers, each layer is defined by a set of points the printer must pass through to produce the desired geometry. G-commands generally tell the printer how to move through these points and how much material to extrude while doing so.
2. T-commands for selecting the active tool to print with. These commands appear most often in multi-material or multi-functional part production, where different printheads are required for different portions of the print. Examples of this include printing parts where certain regions require different material properties (varying rigidity, electrical conductivity, etc.), printing multicolored parts, or optimizing print time by using higher flowrate toolheads with large nozzles for the bulk of the printing and using smaller nozzle sizes in areas where details are required.
3. M-commands for all other printer functionality. On a 3D printer, M-commands most commonly control all heater and fan-related activities, such as reaching a certain nozzle temperature or configuring part-cooling; however, they can also implement more advanced functionality such as updating printer firmware or tuning acceleration profiles to reduce part defects due to printer motion. Although G-commands and T-commands are mostly standard across different commercial printers and slicers, M-commands vary much more and are often used by manufacturers to enable custom functionality specific to certain machines.

Many slicers are available for purchase or to freely download. For example, Cura [94], PrusaSlicer [95] and Slic3r [96] are three powerful, widely used open-source slicers which may be downloaded for free from their respective websites and customized by savvy users to further enhance their functionality. Commercially, Simplify3D is a fast, easy-to-use slicer with a lot of built-in functionality [97]. This is the slicer that we use most often in the LM2. All these slicers can be customized to work with different printers and output standard, human-readable G-code files that may be further manipulated or sent directly to most commercial printers.

3.2.1.3 Steps 3 and 4: Printing parts from a G-code file on a conventional FFF printer

The G-code file outputted by the slicer may be uploaded directly to the printer or sent serially over USB. Assuming the correct filaments and toolheads are installed in the printer and the build surface has been prepared appropriately, this step usually requires very little user-intervention once the print has been started. Exceptions include pausing prints to manually add inserts or other components to the printed parts or dealing with unfortunate printing problems. From a user perspective, this part of the workflow is a single step; internally though, understanding the two main steps the printer executes at this point will make following sections easier to understand.

As shown in Figure 3-1, the printer's main control board receives the G-code file and translates it into actions the printer can execute. Each line of G-code is decoded by the board's processing unit and converted into the appropriate electrical signals required for heating, part-cooling, motion, and extrusion. For example, when the control board comes across the following G1 motion command:

G1 X10 Y15 E2 F2000

It converts it into step and direction signals which are sent to the appropriate onboard stepper-motor drivers to command motor speed and position. This allows the printer to follow specific trajectories while extruding the required amount of material. In this case, the printer moves to the coordinate $(X, Y) = (10,15)$ at 2000 units/min while maintaining its current height in Z and extruding 2 units of filament. The control board executes a similar process when it receives an M-command to turn on a heater, except for the additional fact that it must then continuously monitor that heater's temperature sensor while executing other commands to maintain the correct heater temperature for the duration of the print.

3.2.2 Serial communication overview

To communicate between the printhead controller and the Aerotech and Fanuc motion system controllers, we chose to use serial communication, as detailed in Section 4.2. Serial communication is a data transfer method in which signal pulses, or bits, are sent sequentially over a single wire. These bits are sent according to specific patterns that the receiver can interpret. Different forms of serial communication exist, such as USB (Universal Serial Bus) and Ethernet, two serial protocols commonly encountered in everyday electronics. Serial communication is characterized by its *baud rate*, which determines how fast bits are sent from one device to the other. It specifically indicates the duration of each bit, or how long it stays high or low before the next one is sent. Common baud rates include 9600bps (bits per second), 57600bps, etc.

Serial communication encodes data in *packets*, each of which contains a specific number of bits. Figure 3-2a illustrates how such a packet is structured. Each packet features a *start bit* and one or two *end bits* which surround the data and allow the receiver to know when the data begins and ends. The actual data – usually an 8-bit byte but occasionally longer or shorter – often represents the binary form of a character as standardized by the American Standard Code for Information Interchange (ASCII) and is generally transferred least-significant bit first. The optional *parity bit* indicates if the sum of the data bits is odd or even. Although rarely used, this bit serves as a basic form of error-checking, allowing the receiver to validate that no bits were lost. To better illustrate all these concepts, Figure 3-2b shows the serially transmitted form of the letter 'S' using 8-bit data, no parity bit, one end bit, and sent at 9600bps (referred to as 9600 8N1).

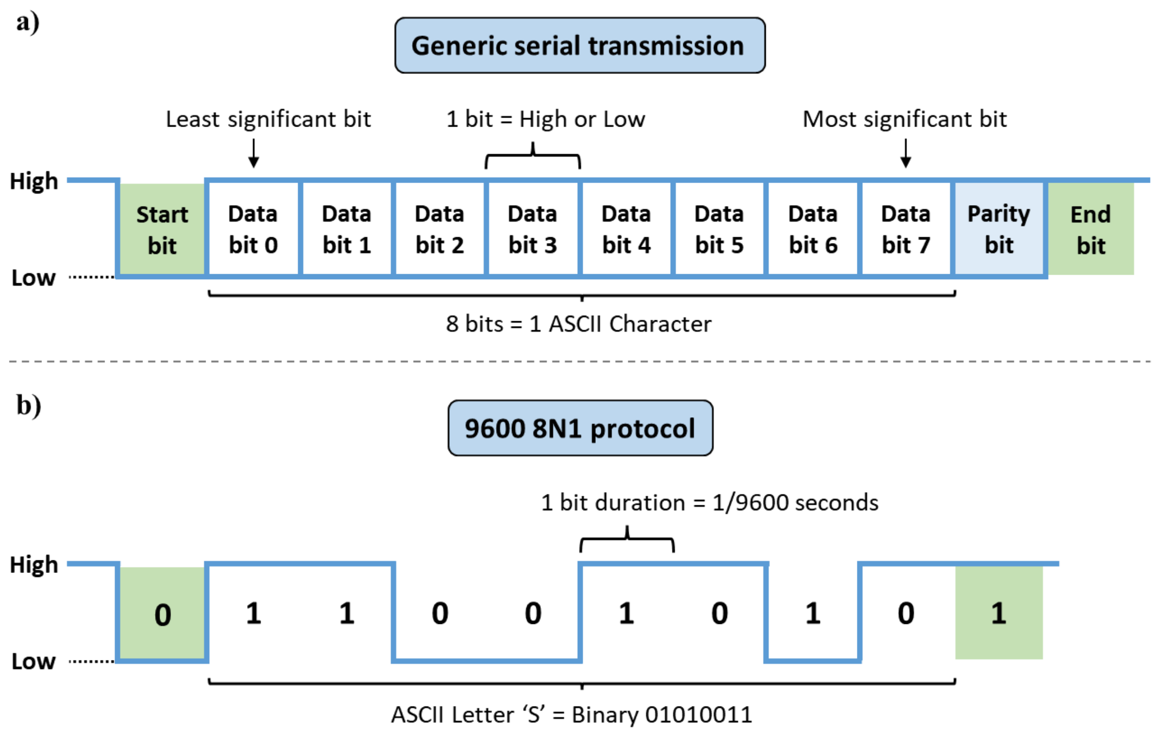


Figure 3-2: a) General form of a packet sent by UART transmission b) Transmitting the letter 'S' using the 9600 8N1 protocol

3.2.3 Common printhead components

To design a truly “universal” controller to drive various types of FFF printheads and even other types of additive manufacturing toolheads, one must first understand what these printheads look like and which elements they share.

FFF printheads typically feature two main regions: a cold end and a hot end. In the cold end, a thermoplastic filament is driven by an extruder at a precise rate. Most consumer extruders feature a motor which spins a set of gears to push the filament towards the hot end. The hot end is separated from the cold end by some form of heat break, usually made of a less conductive material, to reduce heat creeping from the hot end to the cold end. In the hot end, the thermoplastic material is heated up to an exact, user-selected temperature before being extruded through a nozzle and deposited layer-by-layer. Upon exiting the nozzle, the thermoplastic filament is often cooled to make it easier to print certain geometries, or sometimes heated up locally to improve mechanical properties. An E3D V6 hot end [98] paired with a Bondtech BMG extruder [99] is the quintessential example of this arrangement in consumer FFF printing; Figure 3-3 illustrates a section view of this classic printhead extruding a filament, highlighting the main electric components that allow it to do so. We refer to the E3D V6 throughout this document as a baseline example of an FFF printhead.

There exist many variations on this base concept: some printheads can mix different types of filaments in the same nozzle, while others like the Pulsar pellet extruder from Dyze Design forgo the filament entirely and drive thermoplastic pellets through a nozzle using a screw like those seen in injection molding. Nonetheless, because FFF printheads all precisely extrude a base material in the cold end, heat it up in the hot end, and deposit it through some sort of nozzle, they usually feature the same set of components illustrated in Figure 3-3. What differs between printheads is how variations of these components are adapted to a specific application, such as high-temperature printing or large-scale additive manufacturing. Appendix A describes these components in more detail, specifically focusing on electrical requirements and how they interact with an electronic control board. Identifying these requirements allowed us to better choose which control board and other electronics to use in the printhead controller, as explained in Section 5.1.

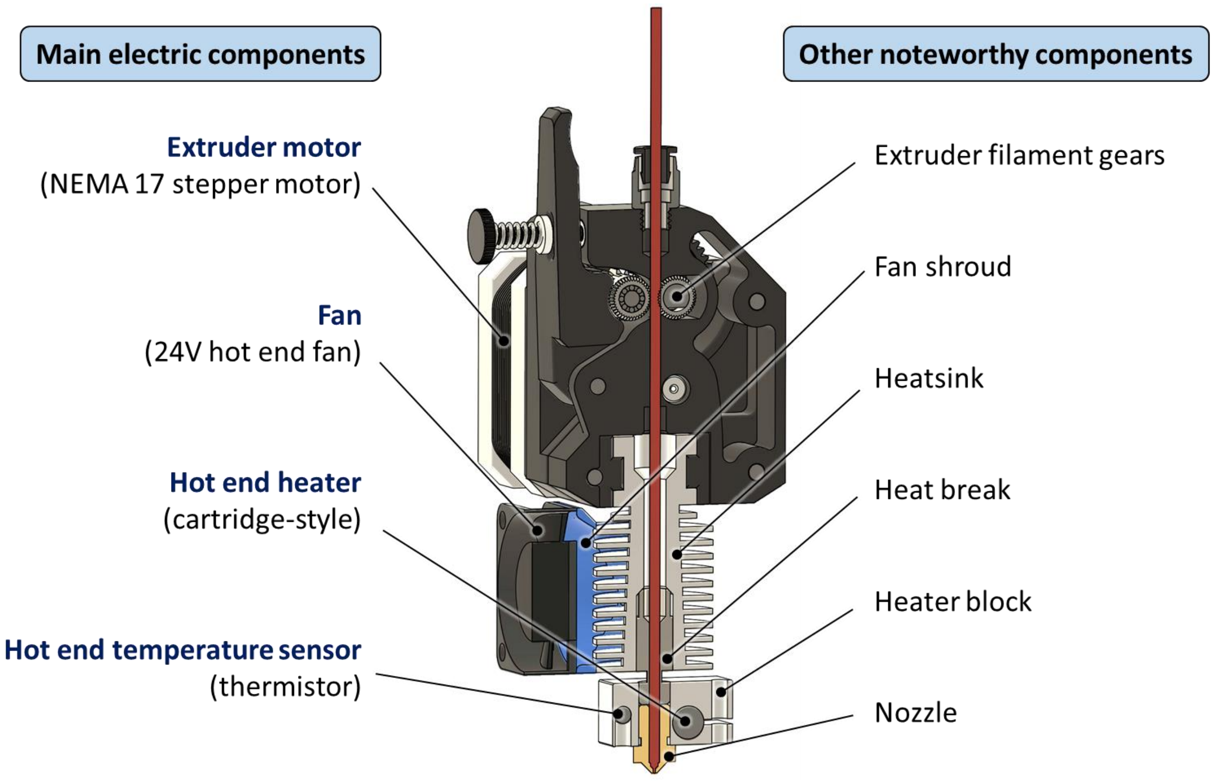


Figure 3-3: Main component overview of a printhead composed of an E3D V6 hotend and a Bondtech BMG extruder (section view)

3.2.4 #3DBenchy benchmark print

One of the most printed benchmark parts in FFF is the #3DBenchy shown in Figure 3-4. This part is designed by Creative Tools and features several complex geometrical attributes to calibrate FFF printers, such as overhanging surfaces and planar or cylindrical surfaces oriented in various directions. This makes it easier to identify which settings to modify to improve overall print quality. The original design is 60mm from front to back, 31mm wide and 48mm tall. [100], [101]

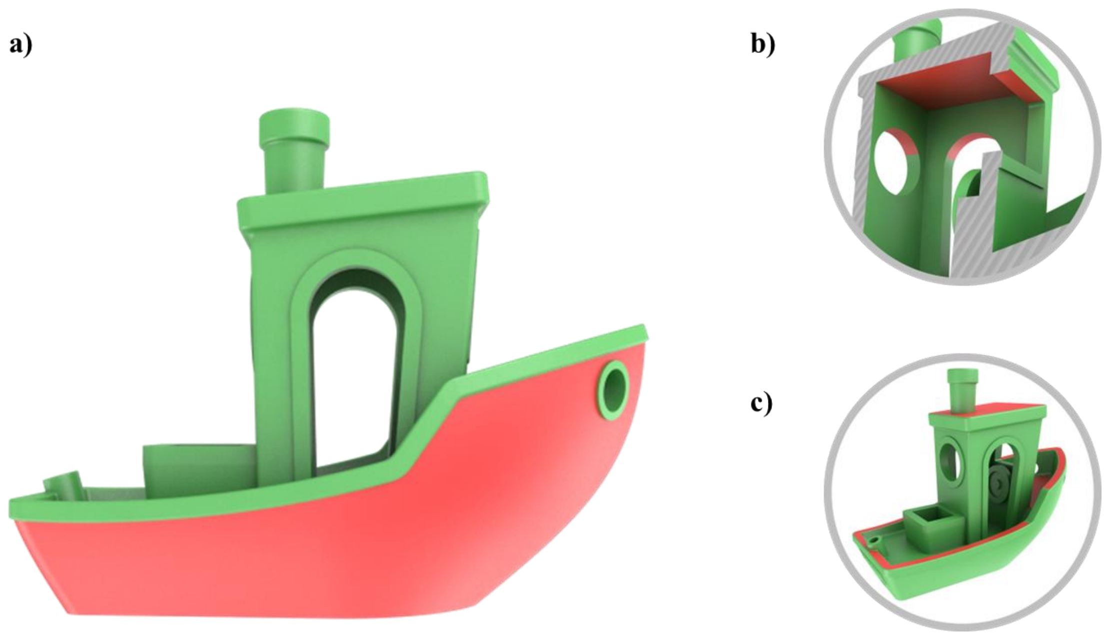


Figure 3-4: #3DBenchy design by Creative Tools highlighting a) large, variably overhanging hull b) various overhanging geometries c) shallow-angled surfaces [101]

CHAPTER 4 DEVELOPING A CUSTOM METHODOLOGY FOR FFF PRINTING ON INDUSTRIAL MOTION SYSTEMS

In Section 2.4, we highlighted the custom workflow that many research groups use to implement FFF on multi-axis motion systems. In this chapter, we describe how we modified the traditional FFF workflow to work with the industrial motion systems in the LM2. This allowed us to complete the project's first sub-objective: developing a custom methodology for FFF printing on industrial motion systems.

4.1 Adapting the traditional FFF workflow to the LM2 motion systems

When compared to the conventional FFF printing workflow presented in Section 3.2.1, additively producing a part on an industrial motion system typically requires a few additional steps. This extra layer of complexity is namely due to two main differences between traditional FFF printers and industrial motion systems:

1. Controllers on industrial motion systems often use a proprietary programming language for motion and auxiliary commands rather than the conventional G-code outputted by most slicers. For example, the XR3 controller used on the Aerotech system in the LM2 is programmed with the AeroBasic programming language; meanwhile, Fanuc robots execute LS or TP program files whose syntax differs significantly from standard G-code. Because of this, to produce FFF parts on an industrial motion system, users must either:
    a. Create their own slicer capable of outputting the appropriate file format for their motion system
    b. Purchase an existing software solution with built-in slicing capabilities for additive manufacturing and adapted to the motion system they use, or
    c. Purchase a software solution capable of translating traditional G-code files into different file formats specific to different motion systems.

This last method is the one we chose for the LM2, as detailed in Section 4.1.2. Because of this, the G-code file outputted by the slicer passes through an additional conversion step before being sent to the motion system controller.

2. Out of the box, industrial motion systems aren't usually capable of 3D printer-specific functionality like extruding filaments or heating thermoplastics; this must be added on by the end user. Doing so often means completing a significant amount of electrical, mechanical, and programming work. It involves:

    a. Selecting a printhead and fixing it to the mobile portion of the motion system

    b. Adding the components required to make this toolhead function correctly, such as heater controllers, water-cooling equipment, part-cooling, motor drives, etc.

    c. Interfacing with the electrical inputs and outputs (I/O) on each motion system controller to command the components selected in the previous step

    d. Writing the code required to drive the I/O selected in the previous step to ensure motion and toolhead commands are executed synchronously.

    This work must be repeated each time a user wants to install a different type of printhead on a motion system. This becomes time consuming in the context of research, where different users regularly switch between a wide variety of printheads for different projects. In the LM2, we decided to use a "universal" printhead controller with multiple input and output possibilities for different toolheads to simplify this process. Doing so allowed us to do most of the electrical integration and programming ahead of time, ensuring compatibility with various types of toolheads and making it easier for end users to focus on printing parts. Using such a controller does create an additional step in the printing workflow, where toolhead commands are transferred from the main motion system controller to the secondary printhead controller, but this step is mostly automatic and requires very little user intervention.

Figure 4-1 illustrates the conventional FFF printing workflow adapted to work with the motion systems installed in the LM2. This workflow was initially developed by Jean-François Chauvette in collaboration with Mëkanic to use their printhead controllers on the Fanuc and Aerotech motion systems. M. Chauvette is a PhD student in the LM2 working on fabricating large-scale abradable geometries on the Fanuc robot using a multinozzle printhead driven by a Mëkanic controller [102]. We adapted each step of this workflow to work with the universal printhead controller we produced in Chapter 5. The following sections describe the workflow in more detail.

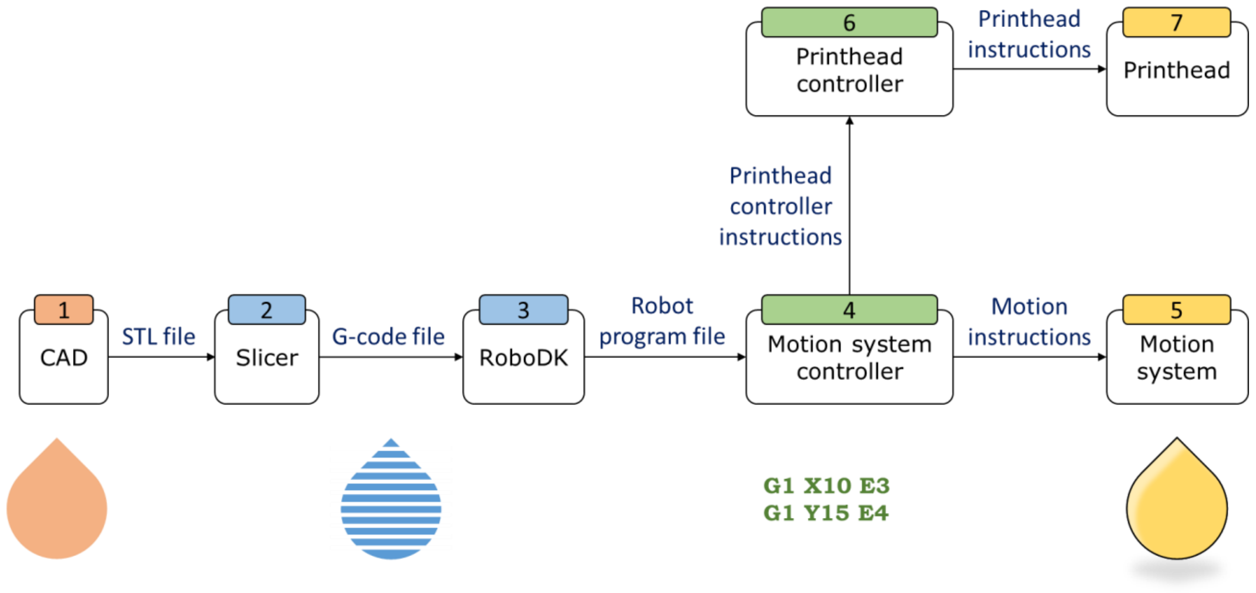


Figure 4-1: Conventional FFF part-production workflow adapted to the LM2 motion systems

4.1.1 Steps 1 & 2: Generating a G-code file from an original part design

Generating an STL file to produce a part on an industrial motion system is the same as in the conventional method. Converting it into a G-code file is also nearly identical, except certain motion systems allow parts to be sliced in ways they couldn't be on conventional printers. For example, the six-axis Fanuc robot in the LM2 can produce non-planar parts, which requires more complex slicing profiles than those produced by a traditional slicer like Simplify3D. Generally, though, the methodology is the same as for a conventional FFF printer: the slicer accepts an STL file and converts it into a G-code file to be dealt with in the following step.

4.1.2 Step 3: Translating the G-code file into an industrial motion system-readable file format

Unlike on a traditional 3D printer, the G-code file outputted by the slicer cannot be sent to the motion system controller directly. A special software must first convert it into the appropriate file format. The LM2 uses RoboDK for this step for several reasons:

1. It can receive a standard G-code file as an input, which is advantageous in many ways:
    a. It allows users to continue using Simplify3D (or equivalent slicers) to slice parts for all the laboratory's printers, facilitating the transition between conventional printers and the Fanuc and Aerotech motion systems, and reducing the number of potential sources of error when comparing parts produced on different machines.
    b. It makes it easier to develop custom tools or integrate existing solutions for slicing parts by building on the standard set of G-code commands and existing slicing methodology. For example, it makes it possible to use open-source G-code post-processing scripts to improve part quality. Additionally, because these tools work with standard G-code, it makes it easier to apply them to other printers and share them with the outside world.
2. It can output printing files for numerous types of motion systems, including the Aerotech and Fanuc motion systems. This means that the same global workflow can be used on systems that are significantly different. If the laboratory decides to purchase a different type of motion system in the future, most of the current printing workflow can be directly adapted to the new machine.
3. It can simulate toolpath motion within the entire workstation ahead of time to ensure no collisions occur. Being able to visualize a print before starting it is particularly important when printing with a six-axis robot arm, where several joint configurations allow the robot to reach the same position in space, but certain configurations may be impossible for a specific application due to cabling restrictions, for example.
4. Most importantly, the software can be customized by users to enable different types of printhead functionality. Because motion systems typically aren't inherently capable of basic

printing functions like extruding material, even simple extrusion and heater-related G-code commands must first be converted into I/O commands that the motion system controller is capable of executing. RoboDK possesses user-customizable Python post-processors to translate G-code into the appropriate file format for a specific motion system, which makes this printhead-function-to-I/O-command conversion reasonably easy to implement. Once the post-processors have been configured, they work in the background of the software and don’t require any additional user-intervention.

Once the correct Python post-processor has been programmed properly, using RoboDK to generate a print file is reasonably simple: import the G-code file generated by the slicer in the previous step, ensure the correct toolhead and printing surface have been selected and generate the file. RoboDK decomposes all G-code commands in the initial file, converts them into either movement instructions that are natively supported by the motion system or printhead-specific I/O commands, then outputs one or several files in the proper format. These files can then be sent to the motion system controller. M. Chauvette completed most of the initial programming work required to make this part of the workflow function correctly with Mëkanic controllers.

4.1.3 Steps 4 & 5: Launching a print file on the motion system controller

Step 4 is analogous to step 3 in the conventional FFF printing workflow: once the print has been launched, it ideally requires little user intervention, despite many operations occurring within the controller. Like on a conventional FFF printer, the motion system controller typically goes through the print file one line at a time, decoding the instructions and converting them into the appropriate signals to drive the proper components. Movement instructions are executed directly by the motion system, bringing the toolhead to the requested coordinates at the appropriate speeds; however, unlike on an FFF printer, extrusion commands aren’t dealt with by the motion controller directly but communicated to the secondary printhead controller via configured I/O.

4.1.4 Steps 6 & 7: Controlling the printhead

The printhead controller contains the appropriate components and circuitry to drive stepper motors, fans, and heaters, monitor temperature sensors, and any other functionality typically required on a printhead. It receives printhead instructions from the motion system controller and executes them immediately, converting them into the proper signals to drive the toolhead. Synchronizing printhead commands with system motion is critical in maintaining print quality, as discussed in Section 4.2 and 4.3. Unlike on a conventional FFF printer, extrusion and motion commands are produced by independent controllers, so extra care must be taken to ensure consistent communication between them and reduce delays.

4.2 Implementing serial communication

One of the most critical parts of the workflow presented in the previous section is communication between the motion system controller and the printhead controller. To properly print parts on a motion system using that methodology, its main controller must send instructions to the printhead controller as the system moves through space, telling it when to heat up, extrude, etc. There are several ways of doing this.

One way involves mapping different analog and digital I/O to printhead functionality: for example, turning on a specific digital output on the motion system might tell the printhead controller to begin heating the filament to a specific temperature mapped to another analog output's value. Mëkanic used this method to interface their controllers with the LM2 motion systems, for good reason: it's a simple, robust way of implementing functionality on multiple systems with minimal delays. However, we identified one main drawback to using this method with the universal printhead controller: it requires multiple I/O to implement basic functionality and offers little flexibility in terms of use-case scenarios. For instance, using the previous heater example, if we wanted to drive a second printhead heater at a different temperature – like on the Typhoon extruder – we would likely need to an additional digital and analog output and program this functionality into both the main controller and the printhead controller.

Because one of the main goals behind designing a universal printhead controller was to allow users to print with a wide variety of toolheads, we decided to go another route: sending G-code commands directly to the printhead controller. G-code has a lot of built-in functionality: it revolves around the concept of tools which can each contain several types of components, including motors, heaters, sensors, etc. Several motors or heaters can be associated with a single tool and turning them on and off is as simple as sending a T-command to activate the correct tool and then sending the appropriate G-code or M-code command. Moreover, because slicers already generate G-code, instead of translating each G-code command into the appropriate I/O signals during the RoboDK post-processing step, we could send many commands to the printhead controller without modifying them. Using G-code thus significantly reduces the amount of programming needed to set up a new printhead.

To send G-code commands to the printhead controller, we decided to use serial communication (see Section 3.2.2 for an overview on this topic). Many microcontrollers – including the Duet 3 mainboard which we selected to drive the printhead controller, as described in Section 5.2.1 – have universally asynchronous receiver/transmitters (UARTs), specialized circuits which allow them to serially send and receive data. These typically use either 3.3 or 5V TTL (transistor-transistor logic) serial – the Duet 3 control board uses 3.3V TTL levels. TTL is based off circuits which use bipolar transistors to switch and maintain logic states and specifies exact thresholds to represent low and high bits. Practically, this means that a low bit is represented by voltages close to 0V and a high bit by approximately 3.3 or 5V depending on the controller. Other hardware implementations of serial communication like RS-232 rely on the same fundamental concept of data transmission but use different voltage levels to represent high and low bits.

Because neither the Aerotech nor the Fanuc motion system implement UART TTL serial communication, we spent a significant amount of time programming them to be able to send G-code commands to the printhead controller. Most of the work referred to in the next two sections was done in collaboration with Abraham Bherer-Constant, who had previously developed the printing methodology to print parts on the Aerotech system and programmed its RoboDK post-processors for different toolhead controllers. We worked with M. Bherer-Constant to create a post-processor for the universal printhead controller on the Aerotech gantry system and to implement serial communication on the Aerotech system and Fanuc robot.

4.2.1 Serial communication on the Aerotech motion system

On the Aerotech system, we chose to send G-code to the printhead controller by using the Aerotech controller's user-configurable position-synchronized outputs (PSO), essentially developing a UART-compatible transmitter from scratch. Although there most likely exists a better, standardized way of implementing such functionality, using these outputs seemed like the most direct way of outputting G-code commands at the time. The PSO outputs on the Aerotech system aren’t normally intended for serial communication, but we found a way to adapt them for this purpose, as detailed in Appendix B.

After solving the various issues that we encountered while debugging (see Appendix B), we finally managed to transmit G-code commands serially. We optimized several parameters to achieve the most reliable transmission possible, which we validated by sending hundreds of G-code strings to an Arduino MEGA microcontroller. By displaying the results on a screen and searching for missing characters or other possible transmission errors, we were able to produce accurate results at baud rates of up to 100 000bps. This baud rate corresponds to 10 000 ASCII characters sent to the Duet control board per second.

Although we can successfully transmit G-code commands from the Aerotech controller to the printhead controller, there are a few limitations to the solution we developed, as described in Appendix B. The most important limitation is that the Aerotech system can serially transmit commands to the printhead controller but cannot receive responses in the same way. This limitation is a minor inconvenience, as digital or analog signals may be used for this purpose instead.

4.2.2 Serial communication on the Fanuc robot

Transmitting G-code commands to the printhead controller over serial was much simpler to implement on the Fanuc robot, which features a user-programmable RS-232 port. RS-232 is a standard protocol for serial communication. RS-232 signals are like the TTL-level signals used by the Duet 3 mainboard in that they contain packets of data surrounded by start and stop bits and sent at specific baud rates; they only differ in the voltages used to represent this data. TTL serial operates between 0 and 3.3 or 5V, with a 0V signal representing a logic low bit and vice versa. On the other hand, RS-232 operates between -25 to -3V and +3 to +25V, with a negative signal indicating a logic high bit and a positive signal representing the opposite. The differences between the two signals are illustrated in Figure 4-2, which shows the ASCII character ‘S’ being transmitted. Luckily, standard circuit boards exist to convert one type of signal to the other. Therefore, as mentioned in Section 5.2.2, we simply installed one such board – a MikroElektronika MIKROE-602 converter – in the printhead controller. This converter allows it to receive RS-232 signals from the Fanuc robot and convert them internally into signals that the UART on the Duet 3 mainboard can understand.

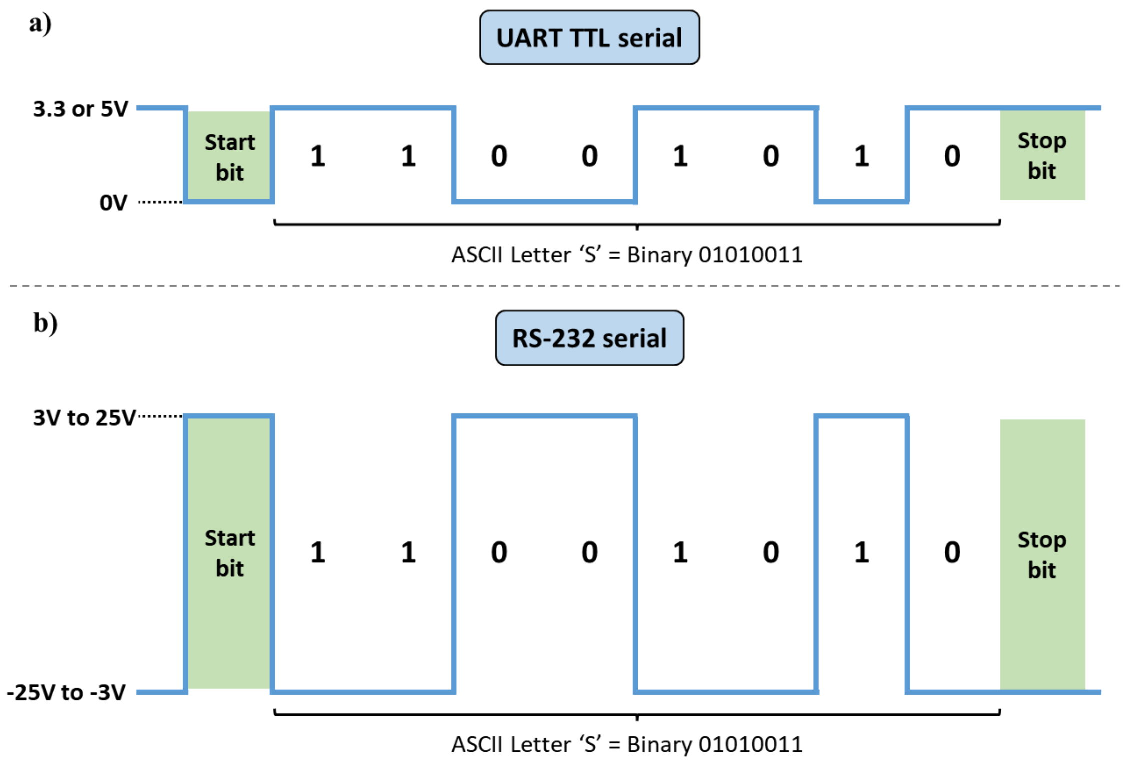


Figure 4-2: Difference in voltage levels between a) UART TTL and b) RS-232 to serially transmit the ASCII letter 'S'

Next, we focused on how to interact with the RS-232 port on the Fanuc. We discovered that the only way to use the port to send or receive signals is to interact with it using Karel programs. Karel is Fanuc's proprietary programming language which allows users to implement a lot of functionality not normally accessible using standard LS files. After speaking with Fanuc, they provided us with a Karel program template to open, read from and write to the RS-232 port on the Fanuc controller, which avoided us the trouble of programming that functionality from scratch. We adapted the template for communication with the printhead controller, configuring parameters like the start and end bits and the baud rate. We then compiled it and uploaded it to the Fanuc controller. To use the RS-232 port, standard LS programs can call upon the Karel program with a character string as an input parameter. The Karel program then writes the string to the RS-232 port, sending it to the printhead controller. To validate that the RS-232 port was transmitting the correct data, we sent a few characters to the port and intercepted the output signals using an oscilloscope.

Like on the Aerotech, there are certain limitations to how we implemented serial communication on the Fanuc robot. Most importantly, although the physical RS-232 port is capable of baud rates of up to 115 200bps, Karel programs limit this value to 9600bps, which prevents us from sending more than 960 characters per second. With standard G-code commands sent to the printhead controller ranging between 2 and 25 characters long, this means sending a single command can take up to 26ms. For time-critical operations like starting extrusion at a specific point in time, this delay may lead to print defects. There is one other main limitation to how we have implemented RS-232 functionality so far: we only ever write to the RS-232 port. Like on the Aerotech motion system, this means the robot doesn't act on responses sent by the printhead controller. Unlike on the Aerotech system, the Fanuc controller's RS-232 port is capable of two-way communication; future users may implement such functionality if required.

4.3 Dynamic extrusion speed control

Transmitting G-code instructions serially is a versatile way of commanding various printhead actions, such as setting heaters to specific temperatures, turning on fans, etc. However, sending conventional G-code commands alone isn't sufficient to accurately control extrusion while printing on an industrial motion system. As the motion system moves from one location to another, it accelerates and decelerates dynamically; the extrusion flowrate must be adjusted accordingly to produce accurate geometries. To achieve this, the motion system's velocity must be transmitted to the printhead controller in real-time for it to adjust the printhead's extrusion speed dynamically.

To implement dynamic extrusion speed control, we decided to replicate the control method developed by Jean-François Chauvette that we had already been using successfully with Mëkanic's FFF controllers on the LM2 motion systems. Due to time constraints, we only implemented dynamic extrusion control for the universal printhead controller on the Fanuc robot, but the same methodology can be easily transferred to the Aerotech system as well. The main difference between this extrusion control method and G-code extrusion commands used on conventional printers is that it regulates extrusion speed rather than the amount of material to extrude. To understand how the method works, consider a typical FFF print, which contains printing sequences – periods of time where the printhead deposits material as it moves through space – and travel moves, where the printhead stops extruding and travels to a new location before beginning to print again. Printing sequences are defined by multiple printing segments, each of which is represented by a G-code command like *G1 X10 Y15 E2 F2000*, as we saw in Section 3.2.1.

There are several parts to the dynamic extrusion control method we implemented. Figure 4-3 shows the transmission architecture which enables dynamic extrusion speed control on the LM2 motion systems. It indicates the outputs that the motion system controller uses to send printhead instructions to the universal printhead controller. We describe each of these outputs in more detail below.

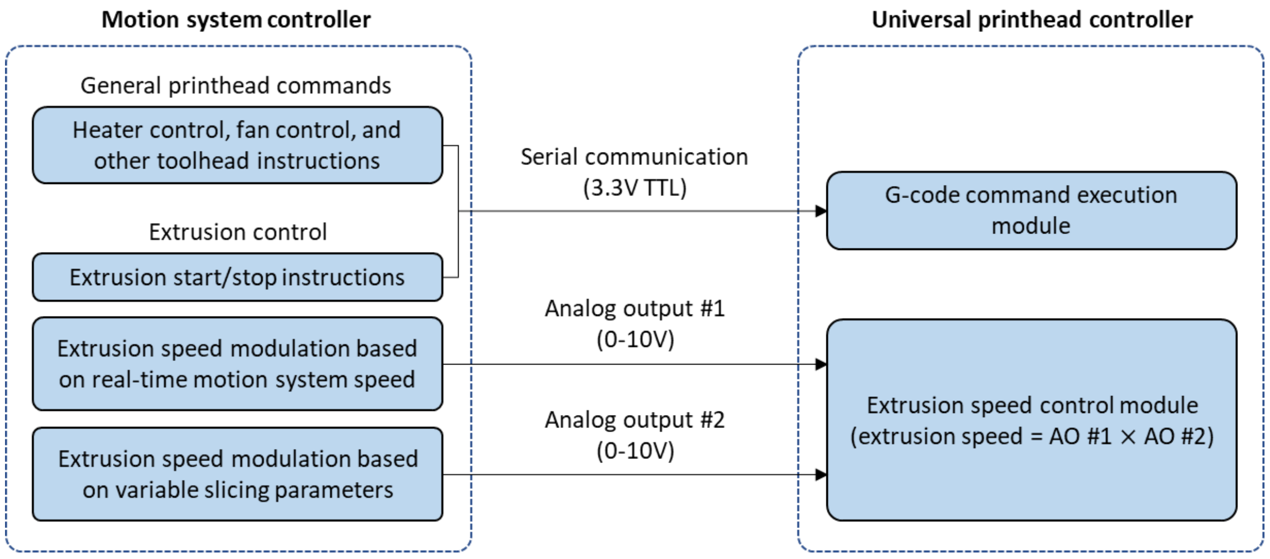

Motion system controller
Universal printhead controller
General printhead commands
Heater control, fan control, and other toolhead instructions
Extrusion control
Extrusion start/stop instructions
Extrusion speed modulation based on real-time motion system speed
Extrusion speed modulation based on variable slicing parameters
Serial communication (3.3V TTL)
Analog output #1 (0-10V)
Analog output #2 (0-10V)
G-code command execution module
Extrusion speed control module (extrusion speed = AO #1 × AO #2)

Figure 4-3: Printhead instruction transmission architecture from the motion system controller to the universal printhead controller enabling dynamic extrusion speed control

First, rather than instructing the printhead controller to extrude the specific amount of material specified by each G1 command, the main motion system controller simply tells it when to begin and end extrusion. At the beginning of a printing sequence, the motion system controller sends a 'start' signal to the printhead controller; when it reaches a travel move instruction, it sends a 'stop' signal to end extrusion. Turning a digital output pin on and off on the motion system is one easy way of doing this; alternatively, sending serial G-code commands can do the same job. We chose the second option for the universal printhead controller to reduce the number of electrical connexions between it and the motion system controllers. The printhead controller deals with the start and stop instructions – which we mapped to the M594 command – like it would with any other G-code command.

Knowing when to start and stop extruding is only part of the puzzle; managing extrusion speed is the most important aspect. Some form of output signal must thus be sent to the printhead controller to modulate extrusion speed in real time. To vary the extrusion speed dynamically, we decided to use not one, but two analog outputs on the motion system controller. These work together to manage different sources of extruder speed variation. Although using two outputs isn't strictly required, the following paragraphs explain why we chose to implement speed control this way.

First, understanding how Simplify3D [97] generates G1 instructions is critical to linking extrusion and motion speed. As illustrated in Figure 4-4, the amount of material required to deposit a filament segment over a specific distance is purely dependent on a few slicing settings specified by the user. As shown in Figure 4-4, we observed that Simplify3D approximates the extruded filament's cross-section by a rectangle whose height l and width w respectively correspond to the layer height and extrusion width specified by the user when slicing their parts. Assuming that the deposited filament volume is equal to the amount of input filament displaced by the extruder gear leads to Equations 1 and 2, which link the extrusion distance E and extrusion speed $F_{extrusion}$ to the travelled distance d and printing speed F_{print}:

$$\frac{\pi}{4}D^2 \times E = lw \times d \Longrightarrow E = \frac{4lwd}{\pi D^2} \tag{1}$$

$$F_{extrusion} = \frac{4lwF_{print}}{\pi D^2} \tag{2}$$

where D is the filament diameter, l is the layer height and w is the extrusion width.

Simplify3D uses Equation 1 to determine how much material to extrude for each G1 motion segment. For a large portion of a typical FFF print, all parameters on the right-hand side of the equation are user-defined constants. This means that the default extrusion speed for most of the print is also constant, only varying due to the robot accelerating and decelerating as it changes direction. We therefore map the maximum value of one of the Fanuc's 0-10V analog outputs (Analog Output #1 in Figure 4-3) to the default extrusion speed specified by Simplify3D. This analog signal is sent to the printhead controller specifying the speed at which it should turn the extruder motor. On the Fanuc controller, we linked this output to the robot's end effector's real-time speed, called the *TCP (Tool Center Point) Speed*. By doing this, as the robot accelerates and decelerates, the analog output's value follows, modulating the extrusion speed accordingly.

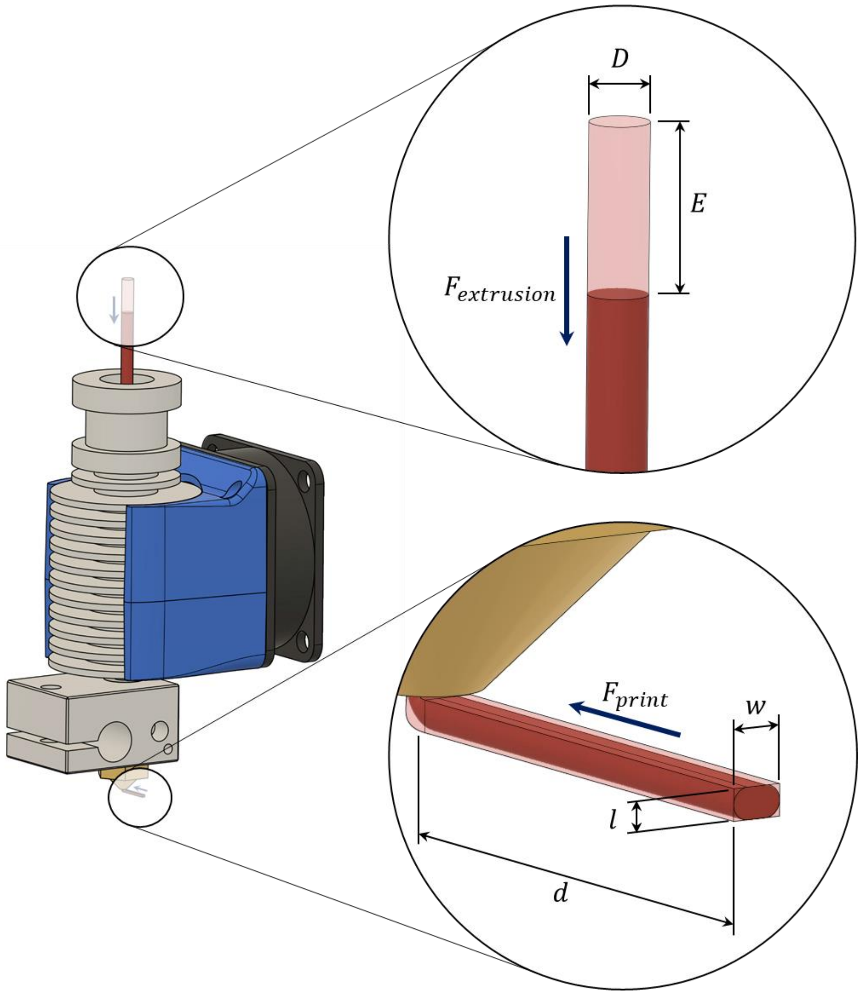


Figure 4-4: Schematic linking extrusion speed and distance to printing speed and deposited filament length as calculated by Simplify3D

For very simple prints, managing extrusion using only two outputs works well; the first output (digital or serial) tells the printhead controller when to start and stop extruding, while the second output (analog) tells it how fast to extrude to deal speed variations. However, this method isn't complete: it assumes that all the parameters on the right side of Equation 2 remain constant for the duration of the print, which isn't usually the case. For example, the first layer of a print is usually printed at a slower speed to increase adhesion to the printing surface; outer part perimeters are often printed slower to improve part surface finish; and different layer heights and extrusion widths are often used for the first layer or even throughout the print. These differences appear in the G1 motion commands outputted by the slicer as different movement speeds or quantities to extrude during certain segments. As a result of this, extrusion speed not only varies due to physical limitations imposed by the motion system, which accelerates and decelerates as it moves through space, but also because of user-defined slicing settings.

To account for this second source of variation, we used a second analog output on the Fanuc motion system (Analog Output #2 in Figure 4-3), called the flowrate correction factor. Normally, the correction factor is set to 100%, which corresponds to default printing conditions. However, when G1 segments require a higher or a lower extrusion speed, we modify the correction factor accordingly. We send this output to the printhead controller, which applies it in real-time to correct the extruder's speed. By multiplying Analog Outputs #1 and #2, the printhead controller determines the real-time extrusion speed needed to deposit material in sync with the motion system displacements.

4.4 Key points

In this chapter, we established the methodology to implement FFF on LM2 industrial motion systems. We especially focused on communication between the motion system controller and the printhead controller, as this link is critical to maintain synchronization between the two systems and produce high-quality parts. The next chapters describe how we built the universal printhead controller described in this methodology, implemented it on the Aerotech and Fanuc motion systems, and used it to print with high flowrate toolheads to produce large-scale parts.

CHAPTER 5 UNIVERSAL PRINTHEAD CONTROLLER DESIGN, FABRICATION, AND INTEGRATION ON LM2 MOTION SYSTEMS

In this chapter, we present the development process for the printhead controller described in the previous chapter. This is the project's second sub-objective: designing, fabricating, and integrating the universal printhead controller on LM2 motion systems.

5.1 LM2 requirements

Before building a universal printhead controller for the LM2, we had to consider a few key requirements. These fall into two categories: printhead and motion system requirements.

5.1.1 Printhead requirements

The main philosophy driving the universal printhead controller's design was maximum compatibility with as many printheads as possible. To achieve this, our goal was to select electronics which could control a wide variety of motors, heaters, temperature sensors, and other common toolhead accessories. We described a few of these common components in Section 3.2.3, with more extensive details about their general requirements provided in Appendix A.

Apart from this general design philosophy, we also identified two specific printheads we knew we wanted to use on the Fanuc and Aerotech motion systems: the Typhoon extruder and Pulsar pellet extruder by Dyze Design. Both these printheads are designed for large-scale, high-flowrate printing. We ensured the controller was specifically able to control the two extruders, whose characteristics are overviewed in the following sections. Figure 5-1 illustrates these printheads, highlighting a few of their main features and components.

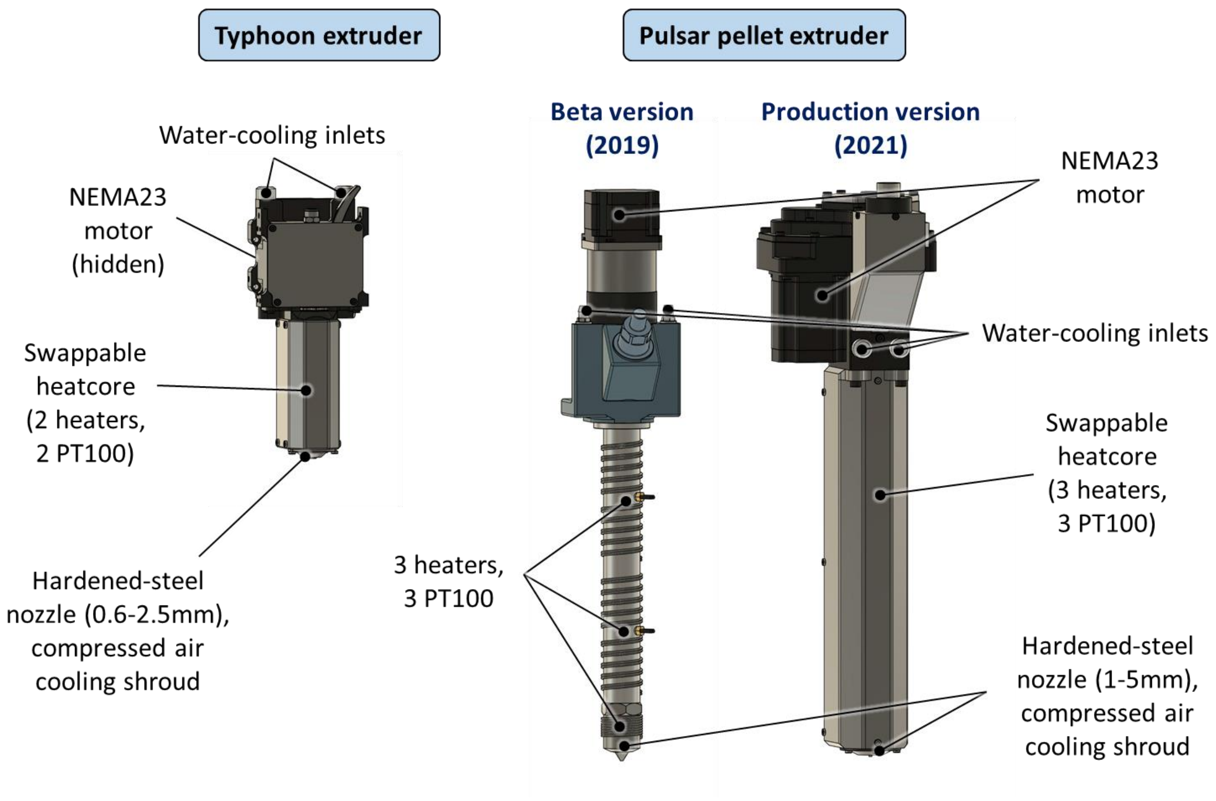

Typhoon extruder
Pulsar pellet extruder
Beta version (2019)
Production version (2021)
Water-cooling inlets
NEMA23 motor (hidden)
Swappable heatcore (2 heaters, 2 PT100)
Hardened-steel nozzle (0.6-2.5mm), compressed air cooling shroud
3 heaters, 3 PT100
NEMA23 motor
Water-cooling inlets
Swappable heatcore (3 heaters, 3 PT100)
Hardened-steel nozzle (1-5mm), compressed air cooling shroud

Figure 5-1: Typhoon extruder and Pulsar pellet extruder (beta version and production version) main feature overview

5.1.1.1 Typhoon extruder

The Typhoon extruder – pictured in Figure 5-1 – is a high-flowrate, all-metal printhead designed and manufactured by Dyze Design. The Typhoon uses a 2.7A, NEMA 23 stepper motor to guide 2.85mm filaments towards the heated portion of the printhead. The entire cold end on the Typhoon is water-cooled which – in combination with its all-metal design – allows it to be used in heated environments to print engineering-grade thermoplastics. This is of great interest to the LM2, since we work with high-temperature thermoplastics like PEEK and PEI on several projects.

The heated portion of the Typhoon extruder, which Dyze Design calls a *heatcore*, features a long, cylindrical metal block that guides the filament towards the nozzle. Two mains-powered coil heaters wrap around the upper and lower part of the block respectively and can be independently set to temperatures of up to 500°C. The heaters require 400W of power. A PT100 RTD sensor in each region monitors temperature. The Typhoon heatcores are shrouded to prevent accidental contact with the hot components and can be quickly swapped without tools, allowing users to quickly change nozzle sizes. The latter are made from hardened steel to prevent deterioration when printing with abrasive materials and range from 0.6 to 2.5mm. In the LM2, we have three heatcores: they respectively possess 0.9, 1.2 and 1.8mm diameter nozzles. Part-cooling is built into the heatcores via nozzle shrouds which guide compressed air towards the printed layers.

5.1.1.2 Pulsar pellet extruder

The Pulsar pellet extruder is also produced by Dyze Design. Unlike the Typhoon and other typical FFF printheads, the Pulsar extruder doesn’t extrude filaments; instead, it uses thermoplastic pellets as a feedstock material, heating and extruding them using a custom screw and barrel assembly like those found in injection molding. This allows it to print at even higher flowrates than the Typhoon extruder and is a more cost-effective way of printing thermoplastics, as pellets are cheaper than filaments. The LM2 initially purchased the beta version of the Pulsar pellet extruder – pictured in Figure 5-1 – in November 2019. The final version, released in 2021 and also shown in Figure 5-1, works very similarly but incorporates improvements identified by Dyze Design during beta testing, like more heating power and a different gearbox design. There are thus slight specification differences between the two units; the following details are for the production version of the Pulsar.

Just like an FFF printhead, the Pulsar extruder possesses hot and cold regions. In the cold end, pellets are guided towards the hopper. A large, 2.7-amp NEMA 23 stepper motor rotates through an intermediate gearbox, pushing the pellets into the barrel and towards the hot end. Like in the Typhoon, the cold end of the Pulsar is water-cooled, preventing the motor from overheating, and ensuring the pellets stay solid at this stage. A capacitive proximity sensor allows the Pulsar to detect when the hopper is empty.

The heated portion of the pellet extruder, which resembles the heatcores on the Typhoon extruder, comprises the barrel, screw, and hardened steel nozzle. Three mains-powered coil heaters allow users to independently heat three different regions to up to 500°C. The heaters’ total power is 1100W. Three PT100 RTD sensors read the temperature in each heated region. On the production version of the Pulsar, heatcores can be swapped reasonably quickly and are available with nozzle sizes from one to five millimeters in diameter.

Table 5-1 summarizes the Typhoon and production Pulsar extruders’ main specifications.

Table 5-1: General specification overview of the Typhoon extruder and production version of the Pulsar extruder

Component	Specification	Typhoon	Pulsar (production)
Motor	Frame size	NEMA23	NEMA 23
	Required current	2.7A	2.7A
Heaters	Number of heaters	2	3
	Operating voltage	120 or 240VAC	120 or 240VAC
	Top heater power	250W	500W
	Top heater recommended fuse rating at 120VAC	2.5A	5A
	Middle heater power	N/A	350W
	Middle heater recommended fuse rating at 120VAC	N/A	3.5A
	Bottom heater power	150W	250W
	Bottom heater recommended fuse rating at 120VAC	1.5A	2.5A
Temperature sensors	Type	PT100	PT100
	Optimal operating temperature range	0 to 500°C	0 to 500°C
Additional sensors	Material presence proximity detector	No	Yes
	Additional connector for a bed proximity sensor	Yes	Yes
	Additional connector for a fan	Yes	Yes
Other	Water-cooling	Yes	Yes
	Compressed air part-cooling	Yes	Yes
	Nozzle size	0.6 to 2.5mm	1 to 5mm
	Maximum flowrate	200mm^3/s	500mm^3/s

5.1.2 Motion system capabilities and requirements

The Aerotech and Fanuc motion systems in the LM2 have distinct capabilities, which means two sets of physical and electrical requirements to consolidate. Table 5-2 highlights a few of the main differences between the two systems relevant to the printhead controller's design. The most noteworthy difference between the two systems is the much larger number of I/O available on the Aerotech system, which makes sense given that it is designed to operate four toolheads at once.

Table 5-2: Comparison of the Aerotech and Fanuc motion system I/O and toolhead specifications

Category	Capability	Aerotech	Fanuc
Physical	Number of toolheads	4	1
Digital I/O	Number of digital inputs	32	16
	Number of digital outputs	32	16
	Digital input voltage	5V to 24V	24V
	Digital output voltage	+24V max.	+24V
Analog I/O	Number of analog inputs	24	4
	Number of analog outputs	24	4
	Analog input voltage range	-10V to 10V	-10V to 10V
	Analog output voltage range	-10V to 10V	-10V to 10V

In addition to the I/O mentioned in Table 5-2, the Aerotech XR3 controllers feature several other input and output connectors, including user-programmable position-synchronized outputs which output 5V signals according to the position of an axis. Section 4.2.1 shows how we used this output to send commands to the universal FFF controller we developed. Meanwhile, the Fanuc robot's main controller features an RS232 port for serial communication and Ethernet ports that work with various industrial connectivity protocols. Section 4.2.2 shows how we used the RS232 port to communicate with the universal FFF controller.

The LM2 motion systems also have more general physical requirements that the FFF controller should respect for ease of use and user safety. Standard 19-inch (482.6mm) racks are installed next to both the Aerotech and Fanuc systems. These hold the various controllers produced by Mëkanic for different additive manufacturing applications, safety equipment, the main XR3 controllers on the Aerotech, and other minor accessories like power strips and shelving. We wanted to make sure the universal FFF controller we developed would fit in these rackmount systems and be easily transportable from one machine to the other. This requires properly routing cables from the controller to the mobile portion of the motion system where printheads are mounted, as well as providing a way to quickly connect and disconnect printheads from the controller.

5.2 Controller design

Once we had identified the above requirements, we searched for electronics we thought could fulfill them in the most flexible way while minimizing the amount of programming and integration work end-users would need to complete.

5.2.1 Control board selection

We first focused on the printhead controller's 'brains': we looked at different control boards that could drive the different components specified previously and whose I/O were compatible with the Aerotech and Fanuc motion systems.

We eventually settled on the Duet 3 Mainboard 6HC produced by Duet3D (component 1 in Table 5-3 and Figure 5-3 below). Duet3D is a UK-based manufacturer of open-source electronics for additive manufacturing and other computer numerical control (CNC) applications such as laser cutting and machining. They produce a diverse range of expandable control boards, expansion boards and other electronic modules mostly targeted at FFF 3D printing. Both their hardware and the software it runs on are open-source and developed using open-source tools. This – combined with their active forum [103], extensive documentation [104], and GitHub pages [105] – makes it easy for users to customize their products to better suit their needs, which is exactly what we were looking for when designing the LM2 universal printhead controller.

5.2.2 Printhead controller main components

Having selected the Duet 3 6HC as our main control board, we then identified which additional components we would need to drive as many printheads as possible with it. One objective at this point was to ideally be able to control four printheads simultaneously so that users could use all four stages on the Aerotech motion system with the same printhead controller. We focused on the Typhoon and Pulsar extruders specifically since future projects might depend on them, but also considered toolheads like the E3D V6 due to its popularity in consumer printing and the standard Dyze Design toolheads we were already using in the laboratory with the Mëkanic controllers (DyzEnd Pro hotend and DyzeXtruder Pro extruder). We ended up selecting the components summarized in Table 5-3. The following sections describe these components in more detail.

Table 5-3: Universal printhead controller component overview and numbering

Number	Component
1	One Duet 3 6HC electronic control board
2	One Mean Well HRPG-600-24 AC/DC converter
3	One E-Switch PA412C1000-136 pushbutton power switch
4	Three solid state relays provided by Dyze Design (equivalent replacement part: Sensata Crydom CWD2425)
5	Seven fuses with different current ratings
6	Two Duet3D PT100 daughterboards
7	One Duet3D PanelDue 5i LCD panel
8	One MikroElektronika MIKROE-602 RS-232 to serial UART converter
9	One SparkFun BOB-12009 logic-level converter
10	One MC74F04N IC inverter (or equivalent)

5.2.2.1 Duet 3 6HC control board (component #1)

The Duet 3 Mainboard 6HC is Duet3D's most powerful control board to date. It contains all the necessary hardware to control stepper motors, heaters, temperature sensors, fans, and other common FFF components. Appendix C outlines its main features in more detail.

5.2.2.2 AC/DC power converter (component #2) and power switch (component #3)

We chose to power the printhead controller at 120VAC so that it could be used with standard outlets installed all over the LM2, including in the racks installed beside the Aerotech and Fanuc motion systems. With standard 120V outlets providing up to 15A of current, this meant 1800W of maximum power for the controller. With the production version of the Pulsar drawing 1100W, this left up to 700W for the rest of the printhead controller. This seemed reasonable given that typical stepper motors used in FFF printing don't require nearly as much power as heaters, standard hot ends like the E3D V6 only require 40W of heating power and other components like temperature sensors and fans have negligeable power requirements. When printing with a high flowrate toolhead like the Pulsar, this does mean that users must be careful not to connect too many other high-power components to the printhead controller, which seems like a reasonable limitation.

To power the Duet 3 mainboard, we chose a 600W AC/DC converter from Mean Well), a reputable producer of power supplies and related equipment. We selected a power supply 100W below the

700W available in case any components drew more power than expected and to reduce the risk of accidentally blowing the 15A fuse. We chose a 24VDC power converter, mostly to maintain compatibility with the small Dyze Design printheads we commonly use in the LM2, which all operate at 24V. Most FFF printing components like heaters and fans operate at 12 or 24VDC, which further justifies this choice.

To turn the printhead controller on and off, we selected an easy to press push-button switch rated to 16A and 125V. The switch provides power to the entire controller, allowing users to quickly turn off all components in an emergency.

5.2.2.3 Solid state relays (component #4)

To drive the Pulsar or Typhoon's heaters at mains voltage, Dyze Design provided solid state relays (SSRs). SSRs are optically isolated relays which function like electromechanical relays but contain no moving parts. When a direct current voltage is applied across the input leads on an SSR, the relay switches on and current flows through the output side. This makes it possible to drive high-voltage, alternating current loads using small, direct current control signals like those outputted by the PWM connectors on the Duet 3 mainboard. We decided to drive only three of the five provided SSRs using the three 2-pin, low-current PWM outputs on the Duet normally intended for driving 2-wire fans. This allows us to use either the Pulsar or the Typhoon with the printhead controller and leaves one high-current and three medium-current PWM connectors for other DC-powered heaters and three 4-pin PWM connectors for fans. The four remaining heater outputs provide ample current for hot ends like the E3D V6; at 24V, the high current output can provide up to 360W while the medium-current outputs each provide 120W, with the V6 only needing up to 40W. This means the printhead controller has enough heater outputs to print with the Typhoon or Pulsar and up to four smaller printheads at once, assuming their total power doesn't surpass the controller's limits.

Although this arrangement means that it isn't possible to use the Typhoon and Pulsar at the same time, it seems like a reasonable choice since both tools fit in the same high flowrate printhead category. If future LM2 users decide they must absolutely use both printheads simultaneously, they can purchase Duet3D expansion boards to obtain additional heater outputs, one of the many advantages of using such a control board.

5.2.2.4 Fuses (component #5)

Dyze Design recommends adding fuses in line with their high-flowrate extruders: two fuses per heater. We therefore added a fuse holder to the controller with spaces for the six fuses required for the Pulsar as well as a seventh 15A fuse for the entire controller to avoid triggering any breakers in the lab if anything were to go wrong. We installed fuses whose ratings would work for both the beta version of the Pulsar and the Typhoon to avoid having to change out the fuses when alternating between the two toolheads. To use the production version of the Pulsar, users may need to change the fuses to better match its fuse ratings; there may be an acceptable compromise which would work for all three high-flowrate printheads.

5.2.2.5 Temperature sensor daughterboards (component #6)

We purchased two Duet3D PT100 amplifying daughterboards to connect to the temperature sensor expansion header on the Duet3, adding four PT100 channels to the existing four thermistor/PT1000 channels on the board for a total of eight temperature sensor inputs. This should provide ample temperature sensor inputs to print with four toolheads simultaneously. We chose not to purchase any thermocouple daughterboards due to their infrequent use in FFF printing and instead maximized the number of available PT100 connections available for our applications. Again, expansion boards make it possible to go back on this decision if LM2 requirements change.

5.2.2.6 LCD Panel (component #7)

We purchased a Duet3D 5-inch PanelDue LCD module which communicates with the Duet 3 control board through a UART (more details about UARTs in Section 4.2) using one of the 9 I/O connectors on the Duet mainboard. The PanelDue allows users to quickly select tools, run G-code commands, turn heaters on and off or extrude material. Like all Duet3D products, its firmware is open source, which means future LM2 users could potentially customize the panel interface to better suit their requirements. For more in-depth control of the Duet 3 mainboard, including modifying configuration files, users can connect to it via any Ethernet-connected computer in the laboratory.

5.2.2.7 Converters for serial communication

To communicate with the Fanuc and Aerotech controllers, we chose to implement 3.3V TTL UART serial communication to send and receive ASCII characters – such as G-code commands – over a single port. This is exactly how the PanelDue interacts with the Duet 3 6HC. Because neither the Aerotech nor Fanuc have TTL connexions designed for this exact purpose, we had to add a few converters to the printhead controller.

5.2.2.7.1 RS-232 to serial UART converter (component #8)

Because the Fanuc has an RS-232 port for serial communication, we added a MikroElektronika MIKROE-602 converter to the printhead controller to convert RS-232 signals to TTL.

5.2.2.7.2 SparkFun logic-level converter (component #9) and IC inverter (component #10)

Although the Aerotech has a few position-synchronized outputs (PSO) which use TTL signals, they operate at 5V rather than the 3.3V the Duet uses. They also idle at 0V when inactive, unlike the Duet UART, which expects the opposite. To compensate for both differences, we used a logic-level converter, which converts 5V signals to 3.3V, and an IC inverter to invert the signals sent from the Aerotech to the Duet 3 6HC.

5.2.3 Enclosure design and connector selection

After identifying the main components required for the printhead controller's operation, we began designing a rack-mountable enclosure to house them. Inspired by Mëkanic's rack-mountable controllers, we looked at enclosures made by Hammond Manufacturing, looking for a size that could comfortably accommodate all the main components with ample room for wiring. We chose the RMCV190513BK1 enclosure, which is approximately 430mm wide, 330mm deep and 135mm tall. We also purchased a removable panel chassis for the unit to mount all the components to, allowing us to easily remove them from the enclosure during fabrication if required. Figure 5-3 in the previous section shows all the components installed in the enclosure.

Choosing enclosure connectors to connect different printheads to the Duet mainboard was the most tedious part of the enclosure design process. Our goal was to make it as easy as possible to connect various types of toolhead components to the controller while maintaining signal integrity and reducing the number and type of cables coming off it. Because these cables would have to then be routed to the mobile portion of the Aerotech and Fanuc systems, we wanted to limit how many cables we would have to deal with, especially on the Fanuc robot where there is little room for cable routing. We spent a long time trying to match connector specifications (available pin configurations, maximum allowable current and voltage) to the Duet control board's various input and outputs requirements. After trying out different connector arrangements in CAD, we finally settled on the arrangement shown in Figure 5-2.

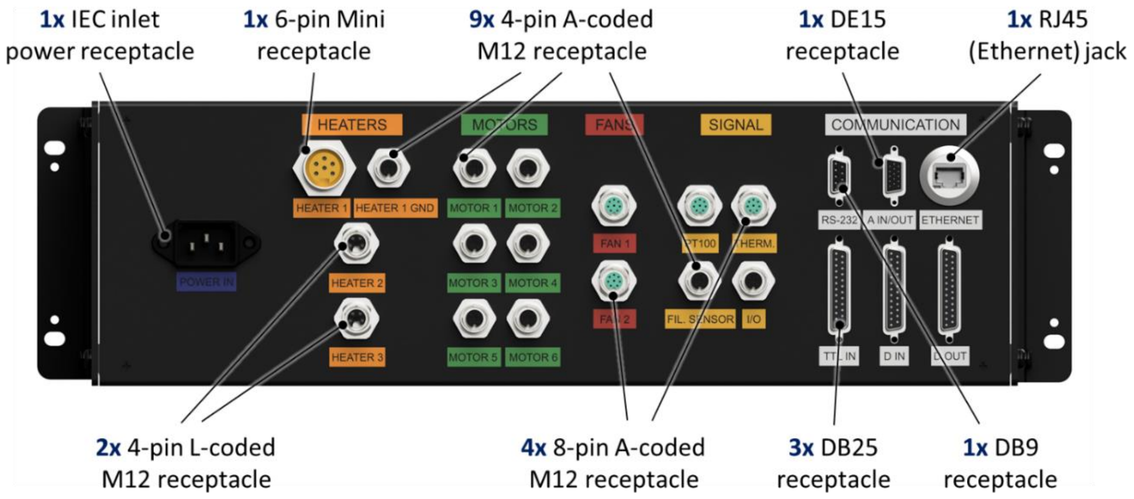


Figure 5-2: Connector arrangement on the back of the universal printhead controller

As shown in Figure 5-2, we chose M12 connectors for most connections due to their reliability and ease of use. These feature a 12mm locking thread and are extremely common in factory automation and other industrial applications. They are available in many different varieties: for instance, we used A-coded 4 and 8-pin M12 connectors for components like motors and sensors that required fewer than 4A of current, and L-coded M12 connectors rated to up to 16A for direct current heater outputs. We were able to connect multiple headers on the Duet control board to each connector, significantly reducing the number of connectors on the back of the enclosure. For example, all four thermistor inputs pass through a single 8-pin M12 connector, as do the PT100 inputs.

To drive high-power heaters using the SSRs in the enclosure, there unfortunately weren't any M12 connectors available with sufficient voltage and current-carrying capabilities, so we used an industrial Mini connector instead. As shown in Figure 5-2, these are like M12 connectors in appearance and functionality, but they’re larger and can carry much higher current and voltage. We used a single 6-pin Mini connector for all three heater outputs.

We chose conventional D-sub connectors for all additional I/O and for serial communication. D-sub connectors are compact, widely available and can easily carry the minimal currents involved in such applications. We installed a standard DB9 connector to communicate with the Fanuc robot using the RS232 protocol, a DB25 connector to receive TTL serial commands from the Aerotech system, and three additional connectors for analog and digital I/O. Most of the pins on these connectors are extra and can be soldered by future users according to their application requirements.

Finally, we selected an RJ45 Ethernet jack to network with the Duet mainboard and a standard three-prong IEC power receptacle to power the entire printhead controller.

5.3 Controller fabrication

Once we were satisfied with the printhead controller’s design, we purchased all the required components and began fabrication. We first machined the openings for the connectors and LCD in the front and back enclosure panels, as detailed in Appendix D. We then proceeded to final assembly and integration.

5.3.1 Enclosure assembly and connector wiring

Once the front and back panels were machined, we installed all the enclosure connectors, LCD, and power button in their final locations. We then assembled the enclosure, leaving the top panel off to allow easier access to components during the following steps. We laid out the components described in Section 5.2.2 on the enclosure’s removable panel chassis and marked their mounting hole locations. We then drilled the holes and attached the main components to the panel which we finally installed in the enclosure, ready for wiring.

To wire up the printhead controller we had to perform over 100 solder terminations and an even larger number of crimp terminations. All the M12, D-sub and Mini connectors on the enclosure’s back panel have solder-cup terminals, while the power plug and power switch have crimp terminals. Inside the enclosure, the Duet 3 mainboard uses crimped Molex-style connectors while the SSRs and power converter use screw terminals. Meanwhile, the serial communication components require soldered connections. Only the Ethernet connector on the enclosure and the PanelDue LCD were plug-and-play, with no crimping or soldering required to wire them up.

After measuring out the required wire lengths in the appropriate gauges, we soldered or crimped each wire to the appropriate enclosure connector terminal on one end, and to the internal component’s terminal at the other, using heat shrink tubing to cover up any exposed connections where necessary. We then completed the remaining internal wiring, connecting the SSRs to the Duet control board and wiring up the power converter and fuse holder. The last wiring step was to connect the power converter and enclosure to the ground wire on the power plug to reduce the risk of electric shocks. Finally, we installed the fuses in the fuse holder and mounted the top panel to the enclosure.

The assembled controller is illustrated in Figure 5-3. Figure 5-3a shows the controller from the front, while Figure 5-3b shows the controller from the top with the upper panel removed to view the internal components. Each component is labeled according to the numbering scheme established in Table 5-3. Figure 5-4 shows the controller in the rack beside the Aerotech motion system, ready for printing.

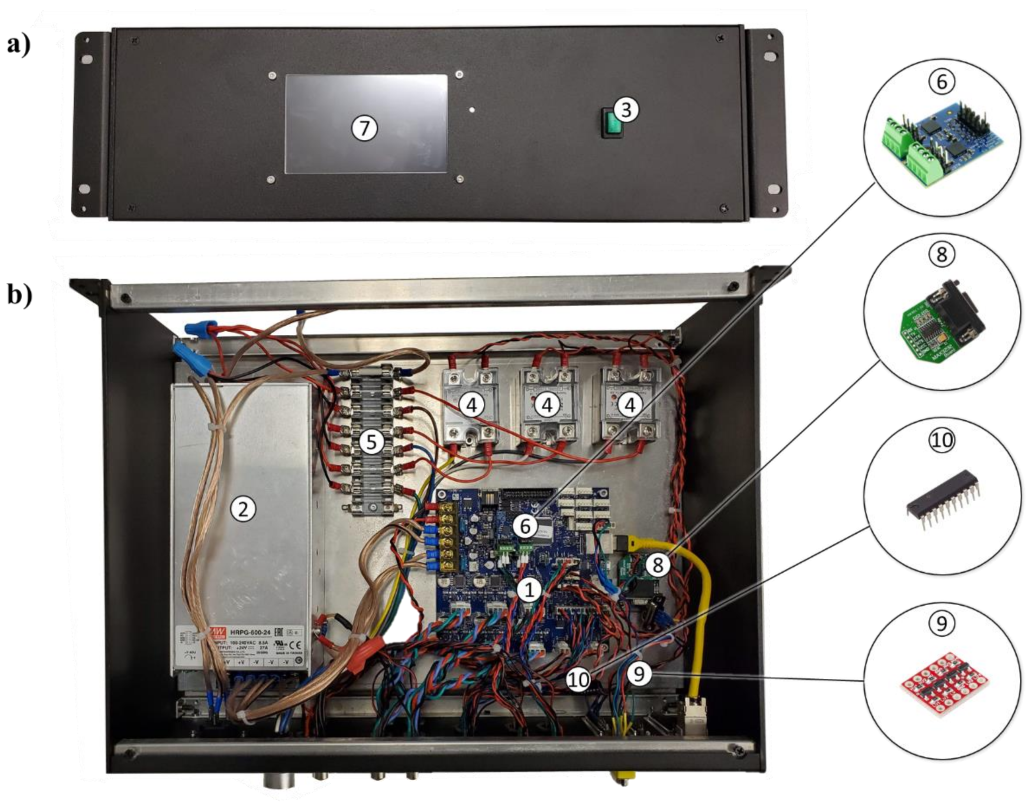


Figure 5-3: Universal printhead controller component overview, numbering, and final assembly and wiring in the enclosure viewed from a) front b) top

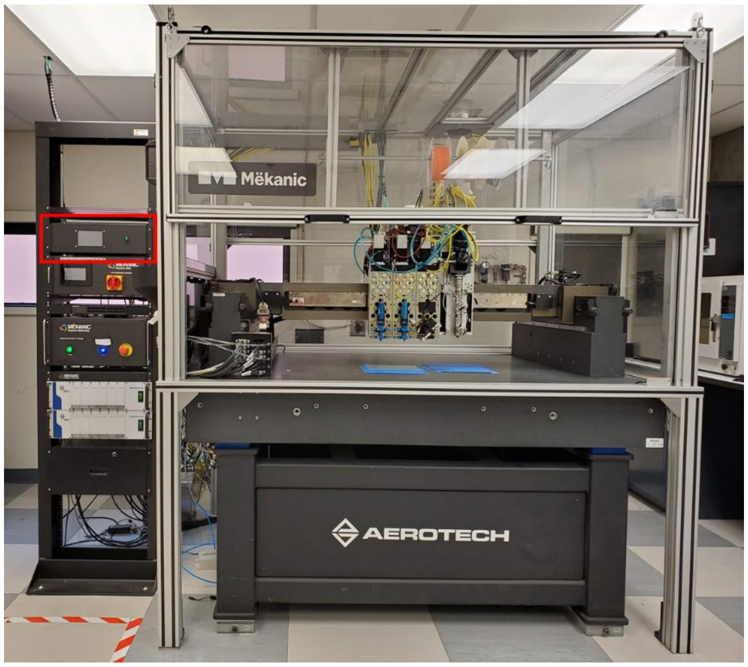


Figure 5-4: Printhead controller installed in the rack beside the Aerotech system enclosure

5.4 Controller implementation on LM2 motion systems

This section describes how we implemented the printhead controller on the Aerotech and Fanuc motion systems. Both systems required the same implementation steps: we first routed cables from the rack-mounted printhead controller to the mobile portion of each motion system; then, we got all connections working properly; finally, we adapted each system's RoboDK post-processor to the new controller.

5.4.1 Cable routing and junction box fabrication

5.4.1.1 On the Aerotech gantry system

To maintain cable integrity and facilitate printhead changes, we chose to replicate the cable routing method implemented by Mëkanic for their controllers on the Aerotech system. We routed cables from the back of the rack-mounted controller to a hole in the left side of the enclosure, up through cable channels to the middle of the enclosure's top panel. As shown in Figure 5-5a, we attached them to the top panel with specialized clips from Igus. Then, we created a controlled cable loop which drops down to a junction box on top of the gantry where users can easily connect and disconnect printheads. We used shielded cables wherever possible to reduce signal noise. Cable shielding is particularly important for low-voltage temperature sensor outputs, which are particularly sensitive to noisy signals like those produced by stepper drivers.

We built the junction box on the Aerotech gantry in a similar way to the entire printhead controller. We selected a self-extinguishing ABS box from Hammond Manufacturing to house the necessary connectors for four printheads installed on the Aerotech stages. We machined openings in the box's sides and installed one Mini and 13 M12 connectors on one side. The junction box's connector arrangement is shown in Figure 5-5b. These connectors reproduce identical connectors on the universal printhead controller, allowing users to interface with the junction box rather than the controller when installing printheads on the Aerotech system. All the controller's connectors capable of driving printhead components are reproduced on the junction box, except two out of six motor connectors; the two additional motor outputs on the main controller can drive rotary axes for print beds or other such systems if users wish to use them.

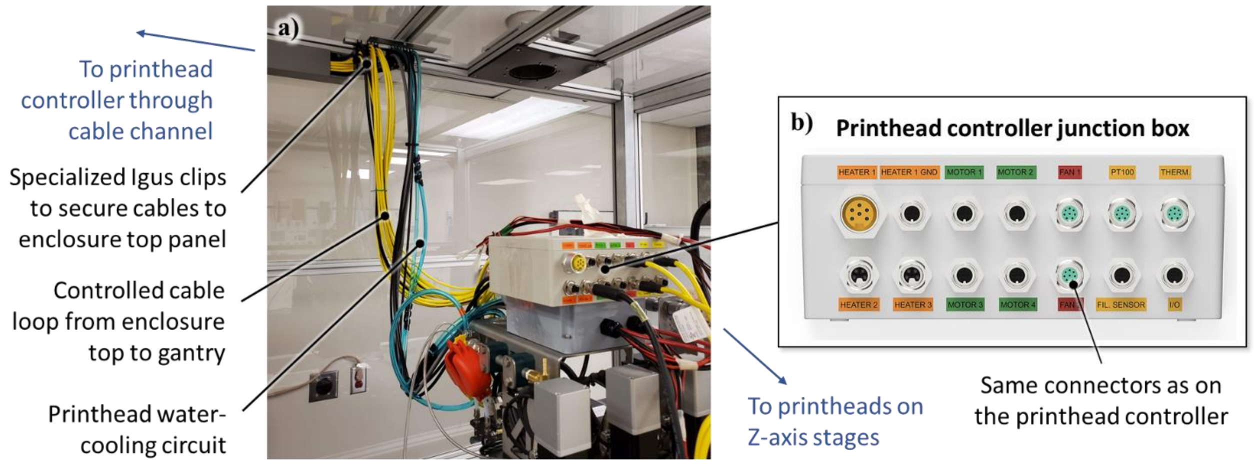

a)
To printhead controller through cable channel
Specialized Igus clips to secure cables to enclosure top panel
Controlled cable loop from enclosure top to gantry
Printhead water-cooling circuit
b) Printhead controller junction box
HEATER 1
HEATER 1 GND
MOTOR 1
MOTOR 2
FAN 1
PT100
THERM.
HEATER 2
HEATER 3
MOTOR 3
MOTOR 4
FAN 2
FIL. SENSOR
I/O
To printheads on Z-axis stages
Same connectors as on the printhead controller

Figure 5-5: a) Universal printhead controller cable routing on the Aerotech gantry system b) Rendered image of the completed junction box

5.4.1.2 On the Fanuc robot

Cable routing on the Fanuc robot was similar to the Aerotech system. We connected shielded cables to the universal printhead controller and routed them to the robot's wrist, where printheads are installed. On the Fanuc system, a specialized multi-axis cable carrier makes this possible while accommodating the robot's large range of motion. Due to the cable carrier's limited size, we had to limit the number of cables we routed to the robot's end effector; while on the Aerotech system we had 14 cables, we could only use seven on the Fanuc robot. The seven connections we chose can simultaneously operate one high-flowrate printhead like the Pulsar or Typhoon extruders and one smaller printhead like an E3D V6. Fewer cables shouldn't be a limiting factor, since the Fanuc robot only has one mobile end-effector, unlike the Aerotech system's four z-stages.

Instead of building a custom junction box like on the Aerotech system, which required CNC machining and several hours of soldering, the cables on the Fanuc robot have connectors at the end of them near the robot's wrist. We plan to 3D print a holder to organize the cables and make them easier to connect to.

5.4.2 RoboDK programming and initial printing tests

Having built the universal printhead controller, and integrated it on the LM2 motion systems, we finally had all the elements required to implement the printing methodology presented in Chapter

4. The final piece of the puzzle was to complete the RoboDK post-processor programming to generate the printing files for the Aerotech and Fanuc systems.

Despite the differences between the two systems, the overall logic that the post-processors use to generate print files is mostly the same. Before post-processing, RoboDK decomposes G-code files outputted by the slicer into a sequence of specific instructions. It treats certain G-code instructions, like M-commands, as a single instruction, while others are subdivided into several instructions, such as motion commands involving extrusion.

Each instruction is categorized as:

1. a motion instruction
2. a speed-setting instruction
3. an M-command call, or
4. an extruder call specifying a specific amount of material to extrude.

The post-processors then sequentially look at each instruction and determine what to do with it. Motion instructions are directly converted into commands that the specified motion system can interpret; for example, on the Aerotech system, G1 motion instructions become LINEAR commands at speeds set by the appropriate speed-setting instructions. We didn’t need to change much in these sections of the post-processors since they had already been configured to work correctly with the LM2 motion systems upon initial setup of the Mëkanic controllers. Instead, we spent most of our time customizing the sections of the post-processors that deal with M-commands and extruder calls. Dealing with M-command calls was easy: because the Duet mainboard interprets G-code directly, we simply outputted instructions to the print file telling the motion system controller to serially transmit the M-commands to the printhead controller. These instructions means that the M-commands from the initial G-code file outputted by the slicer end up being transferred to the printhead controller unchanged.

Dealing with extruder calls required more programming work. At first, we simply transmitted the extruder calls to the Duet controller in the same way as M-commands without implementing the dynamic extrusion speed control method overviewed in Section 4.3. Appendix E overviews this temporary extrusion control method – which we call static extrusion control – and describes many of its limitations. Although using static extrusion control meant the printhead controller didn’t respond to real-time motion system acceleration, the method was much easier to implement. Using this method initially allowed us to quickly troubleshoot the rest of our printing methodology and successfully print our first parts with several different printheads. The goal at this point wasn’t to obtain perfect printing results, but to validate that the printhead controller could drive various printheads and to identify implementation mistakes we might have made.

We tested several printheads at this stage, namely a DyzEnd Pro hotend driven by a DyzeXtruder Pro extruder, an e3D SuperVolcano driven by a Bondtech BMG extruder, the Typhoon extruder, the beta version of the Pulsar extruder, and a custom printhead prototype for printing with continuous fiber-reinforced thermoplastics. Due to machine availability, we tested these printheads on the Aerotech motion system. We then transferred the DyzeEnd Pro/DyzXtruder Pro printhead and the Typhoon extruder to the Fanuc robot. Appendix F and Appendix G show a few of the printheads we tested and the initial printing results we obtained with them.

5.4.3 Implementing dynamic extrusion speed control

Having validated that the printing methodology developed in Chapter 4 was capable of producing parts on both LM2 motion systems, we then integrated our dynamic extrusion speed control method to improve printing results. Integrating the dynamic method required making changes in RoboDK as well as in the Duet 3 mainboard’s firmware. Due to time constraints, we only implemented dynamic extrusion control on the Fanuc robot. Future users should be able to adapt the same method to the Aerotech gantry system by transferring the same methodology to its RoboDK post-processor.

5.4.3.1 RoboDK post-processor modifications

Modifying the universal printhead controller’s RoboDK post-processor to apply the dynamic extrusion control method was reasonably straight-forward, seeing as most of the programming had already been done for the Mëkanic FFF controllers. Most of the work involved modifying the way move instructions were dealt with. Our temporary static extrusion control method involved serially transmitting G1 extrusion commands to the printhead controller based on each move’s average speed; we modified this part of the post-processor to work with the three output signals required for dynamic extrusion control instead.

5.4.3.2 Printhead controller firmware modifications

While modifying the RoboDK post-processor to accommodate the new extrusion-control method was reasonably easy, implementing this new functionality on the printhead controller was much more complex. Out of the box, the Duet 3 mainboard isn’t capable of controlling motor speed based on external inputs. Because it’s designed to control an entire printer rather than just an extruder on its own, this type of functionality isn’t normally something it would need. However, because its firmware is open source – something we prioritized when looking for control boards – we were able to modify it to better fit our application requirements. The entire firmware is written in C++ in Eclipse and our modified version is available on the LM2 GitHub page [106].

5.5 Key points

In this chapter, we developed and integrated a universal printhead controller capable of driving various types of printheads on the Aerotech and Fanuc motion systems. We then used the methodology developed in Chapter 4 to begin initial printing tests on both systems, using a temporary static extrusion control method to simplify the troubleshooting process. Finally, we completed the remaining programing required to implement dynamic extrusion control on the Fanuc robot. The next major milestone to achieve was to print large-scale parts using the methodology and hardware we had developed. We address this topic in the next chapter.

CHAPTER 6 LARGE-SCALE PRINTING RESULTS WITH THE FANUC SIX-AXIS ROBOT

In this chapter, we present our third sub-objective: printing large-scale parts to demonstrate the capabilities of the system we developed in previous chapters. Due to time restrictions and because we had already validated that we could connect various types of toolheads to the universal printhead controller on the Aerotech gantry, we focused on printing with a single printhead on the Fanuc robot. We chose the Typhoon extruder due to its relative simplicity compared to the Pulsar pellet extruder and because we plan on using it for several upcoming projects in the LM2.

6.1 Testing the dynamic extrusion speed control method

To evaluate the dynamic extrusion speed control method's performance, we set up the Typhoon extruder on the Fanuc robot and ran some initial printing tests. We printed the first few layers of a #3DBenchy at 800% scale using different infill configurations to observe the control method in different scenarios:

1. We printed the first three layers of the #3DBenchy with a solid ±45° infill pattern to test how consistently the system could print straight lines, curves, and solid infill. We printed the part using Materio3D 3D850 PLA, a 0.9mm nozzle, 0.4mm layer height, 1.08mm extrusion width, 60mm/s print speed, and 230ºC heatcore temperature. Figure 6-1a shows the result of this first test print.
2. We then printed the ten first layers of the #3DBenchy using the same settings and a basic honeycomb pattern. This allowed us to evaluate if the new method was able to consistently extrude geometries featuring short segments and many direction changes. Figure 6-1b shows the resulting print.

As expected, the printed parts are markedly better than any of our previous results on the Fanuc robot using the static extrusion control method (see Appendix G). The curved segments are smoothly extruded, the solid infill in Figure 6-1a contains few gaps or over-extruded areas, the over-extruded blobs in the corners of both parts are less pronounced than in previous results, and each printing segment is entirely printed, with no gaps at the beginning or end of the segment where the motion system accelerates and decelerates. However, there are still defects in both parts. In

Figure 6-1a, the infill in the top layer features holes in certain areas where the extrusion width randomly drops from the default slicer value of 1.08mm to as low as 0.38mm. The solid infill is over-extruded near the edges of the print where the printhead reverses direction, causing ridges to appear between consecutive raster lines. Similar extrusion inconsistencies appear in Figure 6-1b. Looking at the part up close, the extrusion width on the honeycomb outline isn't constant: it varies continuously as the printhead changes direction, reaching values as low as 0.61mm and as high as 1.87mm.

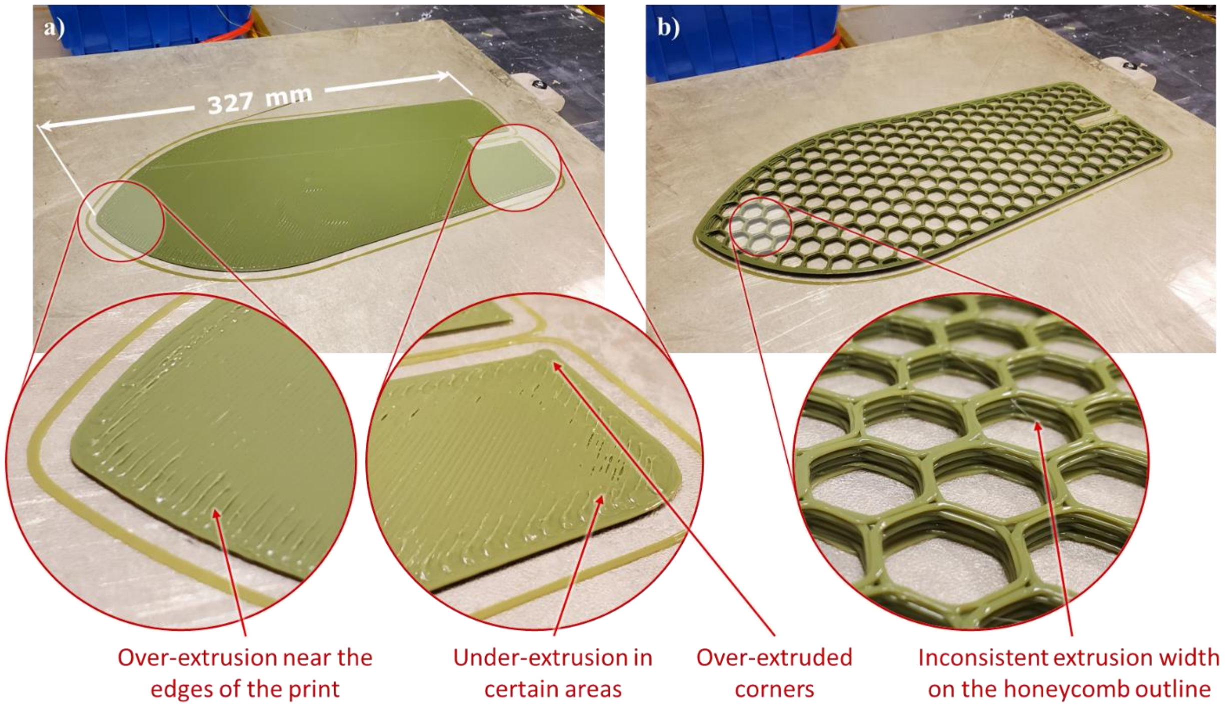

a)
327 mm
b)
Over-extrusion near the edges of the print
Under-extrusion in certain areas
Over-extruded corners
Inconsistent extrusion width on the honeycomb outline

Figure 6-1: a) First three layers of a #3DBenchy printed with the Typhoon extruder on the Fanuc robot using the dynamic extrusion speed control method; solid infill b) First ten layers of the same geometry printed using honeycomb infill

To try to reduce the extrusion inconsistencies we observed in our first two parts, we decided to examine how a Fanuc-specific setting called the *TCP Prediction Time* would affect our printing results. This setting is directly related to the *TCP Speed* analog signal we referred to in Section 4.3 (Analog Output #1 in Figure 4-3), responsible for varying the extrusion speed to match the robot's actual speed. The *TCP Prediction Time* allows users to send this analog signal up to 1 second ahead of time by using the robot's look-ahead capabilities to predict its upcoming speed. In theory, this allows users to compensate for any delays that might cause the extrusion speed to lag the motion system's real-time speed. For our first two test prints, we had used a 100ms prediction time; this was the value that was pre-set for the Mëkanic controllers. Although it worked reasonably well overall, we thought we might be able to find a more appropriate value for the universal printhead controller that would reduce blobbing in the corners and produce more consistent small features like the previous honeycomb geometries. To evaluate the prediction time's influence on these defects, we designed a 90mm by 70mm rectangular plate with regularly placed 8mm-diameter holes in it. We then printed the plate's bottom layer and outline, varying the prediction time between each print. We printed the same part three times, using a 0, 100 and 200ms prediction time, as shown in Figure 6-2a. We used the following print settings: 1.8mm nozzle diameter, 1mm layer height, 1.9mm extrusion width, 50mm/s print speed, and 210ºC heatcore temperature.

The results in Figure 6-2a allow us to draw several conclusions about the TCP prediction time. First, there is excessive stringing in all three parts, as retraction had not yet been implemented at this stage. Second, increasing the prediction time leads to sharper corners, as shown in Figure 6-2b. Figure 6-2b provides the critical dimensions of the top right corner of each rectangular plate. Because the plate is printed with two outlines, one would normally expect the nominal outline width to be 3.8mm, twice the extrusion width. On the actual printed parts, the average outline width is approximately 4.10mm, indicating that the entire part is slightly over-extruded. This kind of general over-extrusion can be easily adjusted by modifying slicer settings for the Typhoon extruder.

Regardless of the overall over-extrusion, one would normally expect the entire outline to measure 4.10mm under optimal extrusion control conditions, including in the corners. However, Figure 6-2b shows that this is clearly not the case. At lower prediction times, there are under-extruded portions at the beginning of every segment as the motion system accelerates, and blobs at the end of each segment. The 0ms prediction time outline width starts as low as 2.99mm at the beginning of the segment, and reaches 8.41mm at the end, more than twice the expected outline width. As the prediction time increases, the blobs and under-extruded areas become less pronounced. The 200ms prediction time produces the best results: the beginning of each segment is no longer under-extruded, although the plate's corners are still slightly over-extruded when compared to the rest of its outline. This phenomenon seems to indicate that there is at least a 200ms delay between the

Fanuc robot’s motion and when the printhead controller receives extrusion commands, which the prediction time solves by sending the commands earlier.

However, the circular profiles show that there is a more complex phenomenon at play. At 200ms, they are extremely under-extruded, which shows that the analog signal becomes unreliable for small, curved features at higher prediction times. At lower prediction times, print quality is much better: some of the circles are fully extruded, with a mostly continuous outline width; others contain gaps or blobs where the printhead begins and ends printing them; but none are as severely under-extruded as at 200ms. Regrettably, choosing the correct prediction time means compromising between corner sharpness and being able to print smaller features. We found that 100ms produced the best results overall but thought we might be able to improve the corner sharpness by increasing the prediction time without reaching the point where the circles were under-extruded. To test this theory, we adapted the rectangular plate so that it featured holes of different sizes – between 2mm and 18mm in diameter – and reprinted it with a 100ms and 150ms prediction time, as shown in Figure 6-2c. Interestingly, despite the print file being identical for both prediction times, the locations of the gaps in the circular outlines vary between the two prints. These results seem to indicate that the prediction time users select impacts not only the timing of the analog signal, but also how accurately it maps to the robot’s actual speed. Chapter 7 discusses this in more detail. Neither prediction time seemed better than the other, so we decided to keep the 100ms prediction time value we had been using originally.

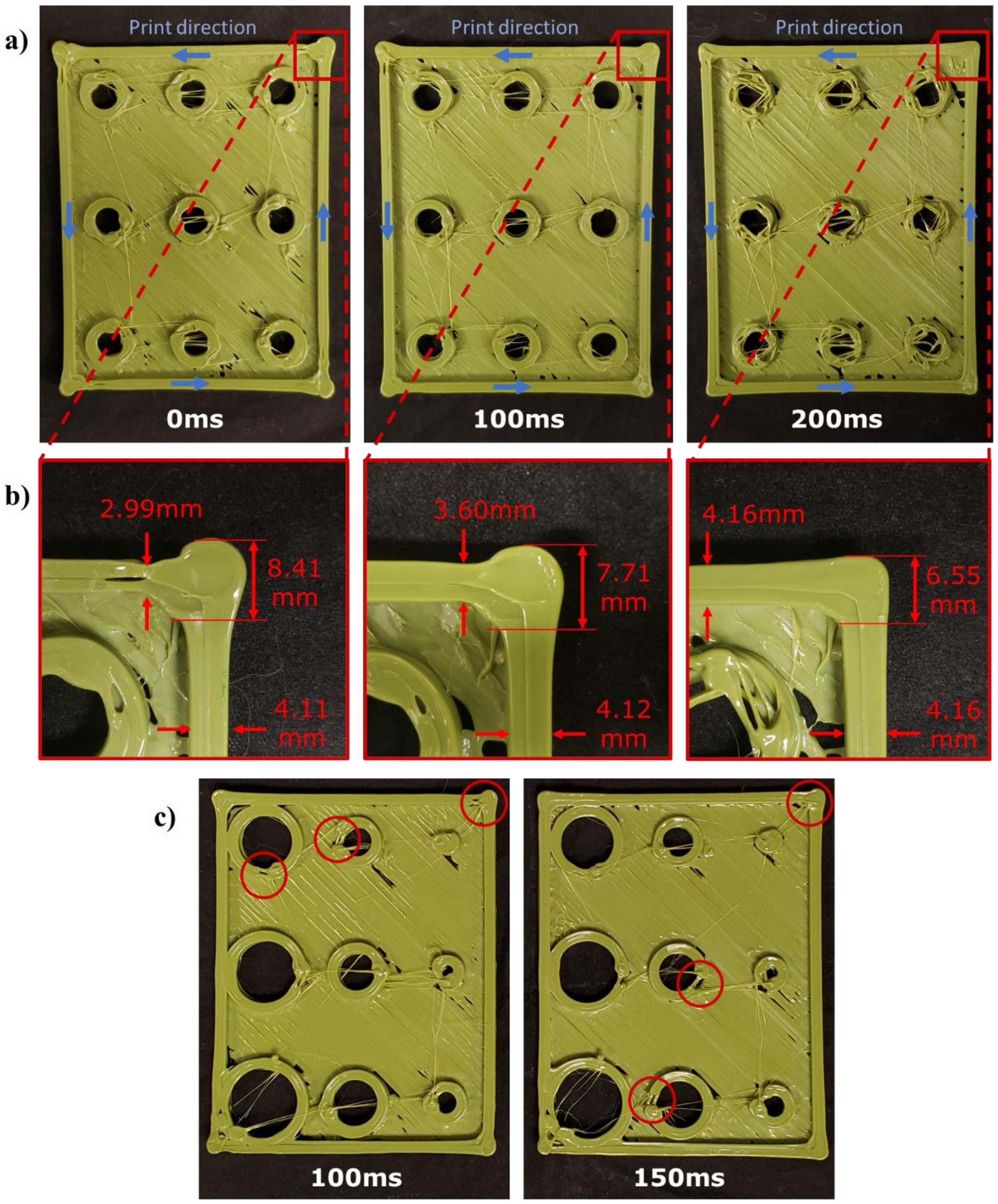


Figure 6-2: a) Rectangular plate printed using three different TCP speed prediction times; effect on corner sharpness and small feature quality b) Over-extruded corner dimensions of rectangular plates printed in a c) Same rectangular plate with variable-diameter holes printed using two prediction times; defect differences between the two parts highlighted

6.2 Printing large-scale parts

Having completed a few test prints to begin debugging the dynamic extrusion speed control method, we moved on to the project’s final sub-objective: printing large-scale parts.

6.2.1 Characterizing maximum extrusion speeds with the Typhoon extruder

Maximum printhead flowrate is a complex phenomenon which depends on many factors. On the mechanical side, motor torque and speed, filament diameter and nozzle size all play an important role. On the thermal side, heater power, maximum temperature, and heater block length are equally important. Finally, material properties like viscosity as a function of temperature are also vital in determining how quickly a printhead can extrude filament. Instead of analyzing each of these properties individually, we decided to analytically determine the maximum extrusion speeds we could achieve with the Typhoon extruder for a specific material and nozzle size.

We characterized the Typhoon’s maximum extrusion speed as a function of printhead temperature as follows: first, we heated up the printhead to a specific temperature within the filament’s advertised printing temperature range and began extruding filament at low speed. We gradually increased the extrusion speed until the extruder motor began skipping steps, indicating it was no longer capable of reliably extruding at the requested speed. We then decreased the extrusion speed slightly, concluding we had reached the maximum achievable speed when the motor could extrude for more than 30s without skipping a single step. Then, we incremented the printhead temperature and repeated the same process. Figure 6-3 shows the results we obtained for 2.85mm-diameter 3D850 PLA from Materio3D using a 1.8mm nozzle. This is the largest nozzle we have in the LM2 for the Typhoon extruder and thus the most suitable for printing large parts rapidly. The secondary axis indicates the maximum flowrates we calculated based on the speeds we obtained.

In practice, the maximum usable extrusion speed to print real parts is slightly lower than the speeds we obtained in Figure 6-3. Although these results indicate when the extruder stepper motor is no longer able to provide the necessary torque and rotation speed to extrude the filament consistently, the actual extrusion flowrate generally decreases below its requested value before this, resulting in poor quality prints. One way to measure at what point this happens is to extrude a specific amount of material at a certain temperature and speed, then to weigh the extruded filament and compare it to the requested value. By gradually increasing the requested speed and weighing the outputted material, one can easily determine when the amount of extruded material begins to drop below the requested amount, indicating that the printhead is struggling to keep up with the requested extrusion speed.

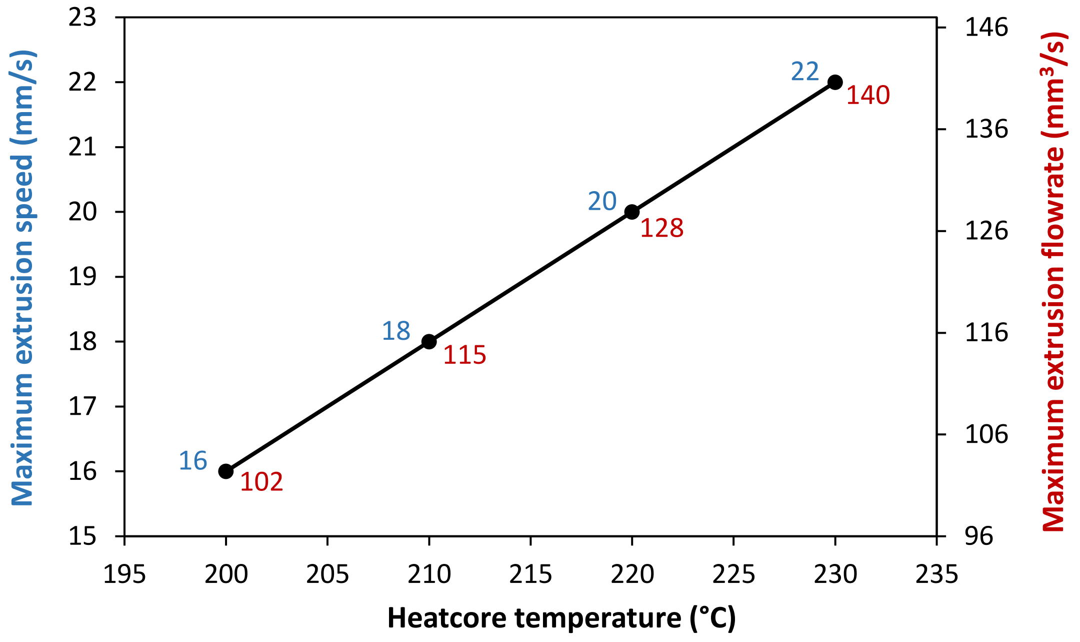


Figure 6-3: Maximum extrusion speed and flowrate for Materio3D 3D850 PLA on the Typhoon extruder using a 1.8mm diameter nozzle

The flowrates we obtained in Figure 6-3 are much lower than Dyze Design's advertised 200mm^3/s maximum flowrate for the Typhoon extruder, but this isn't necessarily surprising. As part of the Typhoon extruder's online documentation, Dyze Design provides specification tables with recommended and maximum flowrates and temperatures for printing with different materials and nozzle sizes [107]. To print Overture PLA (a different brand name filament of the same material) with a 1.8mm nozzle, they recommend printing at 210ºC and 104mm^3/s, with an absolute maximum reachable flowrate of 115mm^3/s. This flowrate perfectly matches the 115mm^3/s we obtained for those settings, as shown in Figure 6-3. Using a larger nozzle size or a different thermoplastic would allow us to print at even higher extrusion speeds; for example, Dyze Design recommends using a 182mm^3/s flowrate to print acrylonitrile styrene acrylate (ASA) with a 2.5mm nozzle.

6.2.2 Printing a large vase

Having identified the maximum extrusion speeds achievable with the Typhoon extruder using Materio3D 3D850 PLA and a 1.8mm nozzle, we began printing large parts. As shown in Appendix G, we had previously printed the beginning of a vase on the Fanuc robot using the Typhoon extruder and the temporary static extrusion control method. The resulting part (shown in Figure G-2b) demonstrated that the Typhoon extruder was capable of printing large parts, but it had very poor surface finish due to the suboptimal extrusion control. We decided to reprint the same part in its entirety this time to compare the two methods and build the LM2's first truly large-scale part produced with additive manufacturing. We printed the part at 250% scale (400mm height compared to the original design's 160mm) using the following settings: 1mm layer height, 2.16mm extrusion width, 60mm/s print speed, 230ºC heatcore temperature, and spiral printing mode (meaning the part is printed as one continuous spiral outline rather than stacked layers). Using these parameters and Equation 2 from Section 4.3, we obtain an average extrusion speed of 20.3mm/s to print the vase; this is lower than the 22mm/s maximum extrusion speed we obtained for 230ºC as shown in Figure 6-3, which indicates that the printing parameters we chose were reasonable from a flowrate perspective.

Figure 6-5a shows the final part, which weighs 640g and took approximately three hours to print. This print time is 67% larger than Simplify3D's projected 1h48m. Simplify3D print time estimates are usually extremely inaccurate, as the software assumes that the entire print is completed at the default print and travel speeds specified by the user, without considering motion system acceleration. Other slicers allow users to input their printer's maximum speed and acceleration as well as a few other motion-specific settings to obtain more accurate results, but Simplify3D doesn't have this capability, which leads to it generally underestimating print times. However, we noticed that the software underestimated Fanuc print times by an even larger factor than with other printers. Future discussions with Fanuc representatives gave us a few hints about why this might be the case; it seems the robot controller's look-ahead capabilities causes the robot to slow down when printing many segments in a short amount of time. Chapter 7 discusses this limitation in more detail.

To estimate how much faster this part production process is compared to traditional FFF printing, we decided to slice the same part with Simplify3D using more conventional settings. According to

Prusa3D’s online knowledge base, the standard E3D V6 nozzle is capable of printing PLA at up to 15mm^3/s flowrates under optimal conditions [108]. Using a standard 0.6mm nozzle and 1.75mm diameter filament, and printing with a reasonable 0.4mm layer height and 0.7mm extrusion width, Equation 2 allows us to calculate a maximum possible printing speed of ~54mm/s. With these settings defined in Simplify3D, we sliced the part such that its outer wall and bottom thicknesses would be the same as the vase we printed with the Typhoon extruder. Simplify3D estimates that using the E3D V6 to print the same part would take 11h12m, more than six times its 1h38m estimate for the Typhoon extruder. Assuming the same 67% increase in real versus estimated printing time as we observed when printing the vase with the Typhoon extruder, this means printing the actual vase with the E3D V6 could take up to 18h42m. These figures show just how much faster we are now able to print large-scale parts on the LM2 motion systems when compared with conventional FFF printheads.

Using the Pulsar extruder with its even greater achievable flowrates would allow an even more substantial reduction in print times, assuming the part geometry is printable using larger nozzle sizes. To estimate how much faster the Pulsar could print the same size vase, we consulted Dyze Design’s online documentation, which recommends using a 2581g/h flowrate and a 200ºC printhead temperature when printing 3D850 PLA with a 5mm nozzle [109]. Given 3D850 PLA’s 1.24 g/cc specific volume [110], this equates to 578mm^3/s, almost six times the maximum flowrate we obtained in Figure 6-3 for the Typhoon extruder using Materio3D 3D850 PLA at 200ºC. Using Equation 2, this means we could print the same vase at 48mm/s if we use a 2mm layer height and 6mm extrusion width. Entering these parameters in Simplify3D yields a 55min estimated print time. Assuming the same 67% increase as before, this means the real print should take about 1h32mins, less than half the print time we observed with the Typhoon extruder.

This improvement becomes even more impressive considering that the vase’s outer walls would be 6mm-thick if printed with the Pulsar, almost three times the Typhoon vase’s 2.16mm wall thickness. Simplify3D estimates the vase would weigh 1845g compared to 640g as printed with the Typhoon extruder. Although this isn’t a problem in this case, it emphasizes one of the weaknesses of using such large nozzles: smaller features are impossible to create, meaning users must select their printhead’s nozzle size as a function of the smallest feature they wish to print. Reducing nozzle size to accommodate small features can have a large effect on overall print times, which is why several groups try to use multiple nozzle sizes to produce a single part, as seen in Chapter 2.

Figure 6-4 compares the spiral vase print times between the Typhoon extruder as observed on the Fanuc robot and the E3D V6 and Pulsar pellet extruder predictions we presented above. The figure indicates two types of print times: those estimated by Simplify3D based on slicing settings, and those observed or predicted based on the 67% increase on the Fanuc robot. The figure highlights just how much faster the system can print using a high flowrate printhead like the Typhoon or Pulsar extruder. It also shows that transitioning from a conventional FFF printhead like the E3D V6 leads to a much greater reduction in print time than moving from the Typhoon to the Pulsar extruder for this specific geometry. This reinforces the point we made above about the importance of matching printhead nozzle size to part geometry: in this case, using a higher flowrate printhead halves print time but leads to significantly more material usage and a heavier final part.

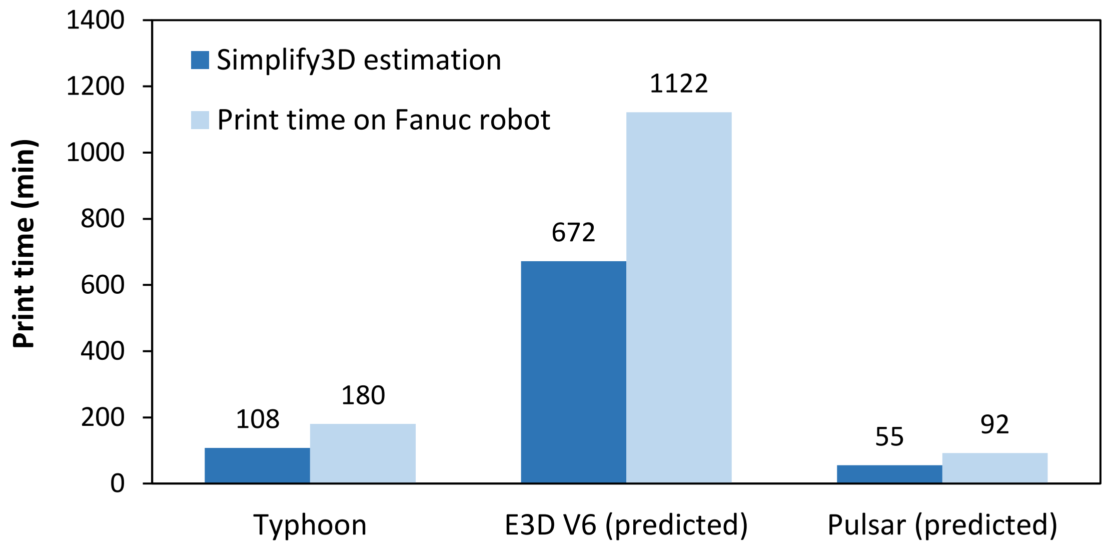


Figure 6-4: Spiral vase print time comparison between the Typhoon extruder, E3D V6 (predicted using Simplify3D) and Pulsar pellet extruder (predicted using Simplify3D)

Figure 6-5b highlights some of the printing defects we observed on the final part. The most obvious printing defect was caused by user intervention: about two-thirds into the print, the extruder finished the first spool of filament we had installed, forcing us to pause the print to install a new one. This interruption caused a hole in the print where the printhead had stopped – circled in red – and a distinct line between the layers printed with the old and new filaments. Apart from this major defect, there are also other smaller problems with the part, including blobs and occasional gaps in the printed outline. Many of these are circled in yellow. On a conventionally printed FFF part, we would normally think that this type of defect is linked to the common start and end point of a printed outline. This area is often the source of such discrepancies in a part’s outline as the nozzle starts or stops extruding and the printhead accelerates from rest or decelerates before changing direction. However, because we printed the vase using spiral mode, each layer smoothly transitions into the next without stopping extrusion; this type of defect is therefore not slicer related.

There are a few possible explanations for such defects in the vase: first, the analog signal sent to the printhead controller might feature sudden variations, causing random fluctuations in the extrusion speed. Second, the error might be in the firmware changes we made to the printhead controller, causing the printhead controller to occasionally stop extruding or over-extrude. Otherwise, the error might not be electronics-related, but caused by pressure gradually building up in the Typhoon extruder and suddenly releasing, creating an over-extruded blob followed by a gap in the print due to absence of material in the nozzle. If this last explanation is correct, tweaking the print settings might solve the issue. Future work is required to identify the exact cause of these defects; using an oscilloscope to monitor the TCP speed analog signal (Analog Output #1 in Figure 4-3) during the entire print would be a good first step. We discuss this in Chapter 7.

Despite the defects we observed in the vase, the fact that we were able to print it successfully is encouraging for future large-scale printing in the LM2. Figure 6-5c shows this vase beside the one we printed in Appendix G using the static extrusion control method. As expected, the dynamic extrusion speed control method produces much smoother curved profiles, eliminating most of the blobs we observed when using the previous method.

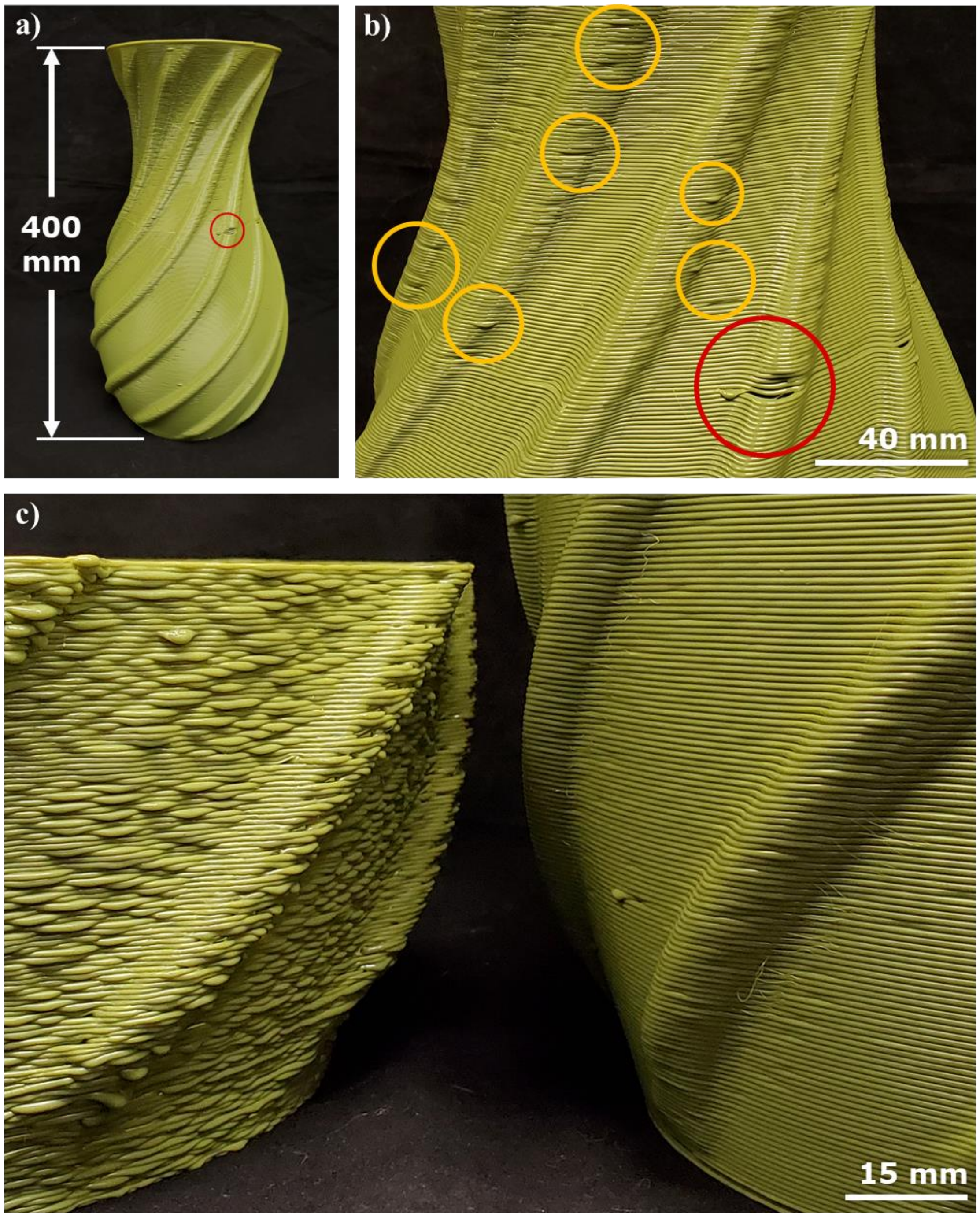


Figure 6-5: a) Spiral vase by JJ76 printed with the Typhoon extruder on the Fanuc robot using the dynamic extrusion control method b) Main defects highlighted, including the spot where the printhead paused during a filament change (circled in red) and other minor defects (circled in yellow) c) Comparing the vase's surface roughness to the one previously printed using the static extrusion control method (see Appendix G)

6.2.3 Printing a large #3DBenchy

After confirming that we could print a large, simple part on the Fanuc system with the dynamic extrusion control method, we decided to move onto a more complex benchmark. Like we had originally done on the Aerotech system using the static extrusion control method (see Appendix F), we printed a #3DBenchy to evaluate what the system was capable of. We printed an even larger, 400% scale #3DBenchy using the following print settings: 1mm layer height, 1.9mm extrusion width, 50mm/s print speed, 210ºC heatcore temperature, and 10% rectilinear infill with 2 perimeter outlines. Figure 6-6a illustrates the #3DBenchy sliced in Simplify3D according to these settings, while Figure 6-6b shows the printed part. Print time was approximately 3h20m (versus 2h00 estimated by Simplify3D, resulting in the same 67% increase as when printing the vase) and the final part weighs 636g.

Like for the vase in the previous section, we had to pause the print near its end to replace the empty filament spool with a new one; having run out of Materio3D PLA spools, we used a slightly different color Ecotough PLA from Filaments.ca. Despite using a new spool from a different manufacturer, we were able to successfully complete the print after changing spools. Unlike with the vase, the seam on the part where the filament swap occurred doesn’t feature any distinct printing defects that aren’t present elsewhere on the part. The new filament also seems to print nearly identically to the first.

Although the #3DBenchy design contains several complex elements, our system managed to complete the entire print, successfully creating many of its challenging features. Despite the Typhoon extruder’s high flowrate, we were able to optimize the printing parameters and cool the deposited material sufficiently quickly to produce steep overhangs and even bridging. Figure 6-6b shows that the Fanuc robot was able to print the boat’s hull without it sagging, even in steeper regions. The system also managed to complete the bridge’s roof successfully despite it’s first layers being printed midair. These results are highly promising, as they show that higher flowrate printheads can be used to print the same complex geometries produced by conventional FFF systems. Using Simplify3D and assuming a 67% real print time increase when using the Fanuc robot, we estimate that printing the same #3DBenchy would take nearly 19 hours with an E3D V6.

Although the final part in Figure 6-6b looks like a #3DBenchy overall, print quality is unfortunately rather poor: numerous printing defects appear all over its outer surface. Like on the vase in the previous section, the outer surface of the part is covered in small over-extruded blobs and gaps in seemingly random locations. These are particularly visible on the outer hull, which should normally be a large, uninterrupted smooth surface. The largest defects appear in the central bridge portion, which feature the smallest details and more frequent direction changes. Here, it seems like the extrusion struggles to follow the robot's motion speed: the shortest segments feature multiple blobs and under-extruded areas. Similar under-extrusion appears after every direction change, causing gaps in the part outline. These extrusion issues are likely caused by the analog signal mapped to the TCP speed not matching the robot's actual speed, as we observed in Section 6.1. We discuss this issue in Chapter 7.

Another issue we observe on the final part is rounded corners in areas which should ideally be sharp. This defect is caused by corner rounding on the Fanuc robot causing it to skip certain points. The robot executes corner rounding to maintain print speed rather than stopping completely at every point to change directions, which would otherwise lead to exceedingly long print times and a reduction in extrusion smoothness. However, because the robot deviates from the trajectory specified by the slicer to do so, users must make a difficult choice: producing accurate parts by completing every segment exactly as planned by Simplify3D or rounding corners to obtain more reasonable print times. The choice becomes even more complicated when we consider that accelerating and decelerating to stop at every point speed likely contributes to extrusion issues due to the TCP speed signal mismatch that we observed on all our parts.

Apart from minor stringing during travel moves, there is one other minor defect on the final part: the smokestack diameter features two distinct lines where the printing speed changes to allow more time for cooling during shorter layers, as shown in Figure 6-6. This geometrical defect is likely caused by the combination of the high flowrate and temperature used to print the part, leading to smaller layers not cooling completely before subsequent layers are deposited onto them. The issue is amplified by the sudden speed changes in these regions. Solving this issue could involve using a lower printing temperature and gradually lowering the printing speed in these regions rather than abruptly doing so from one layer to the next. Although Simplify3D doesn't offer this functionality natively, this could easily be done with a post-processing script.

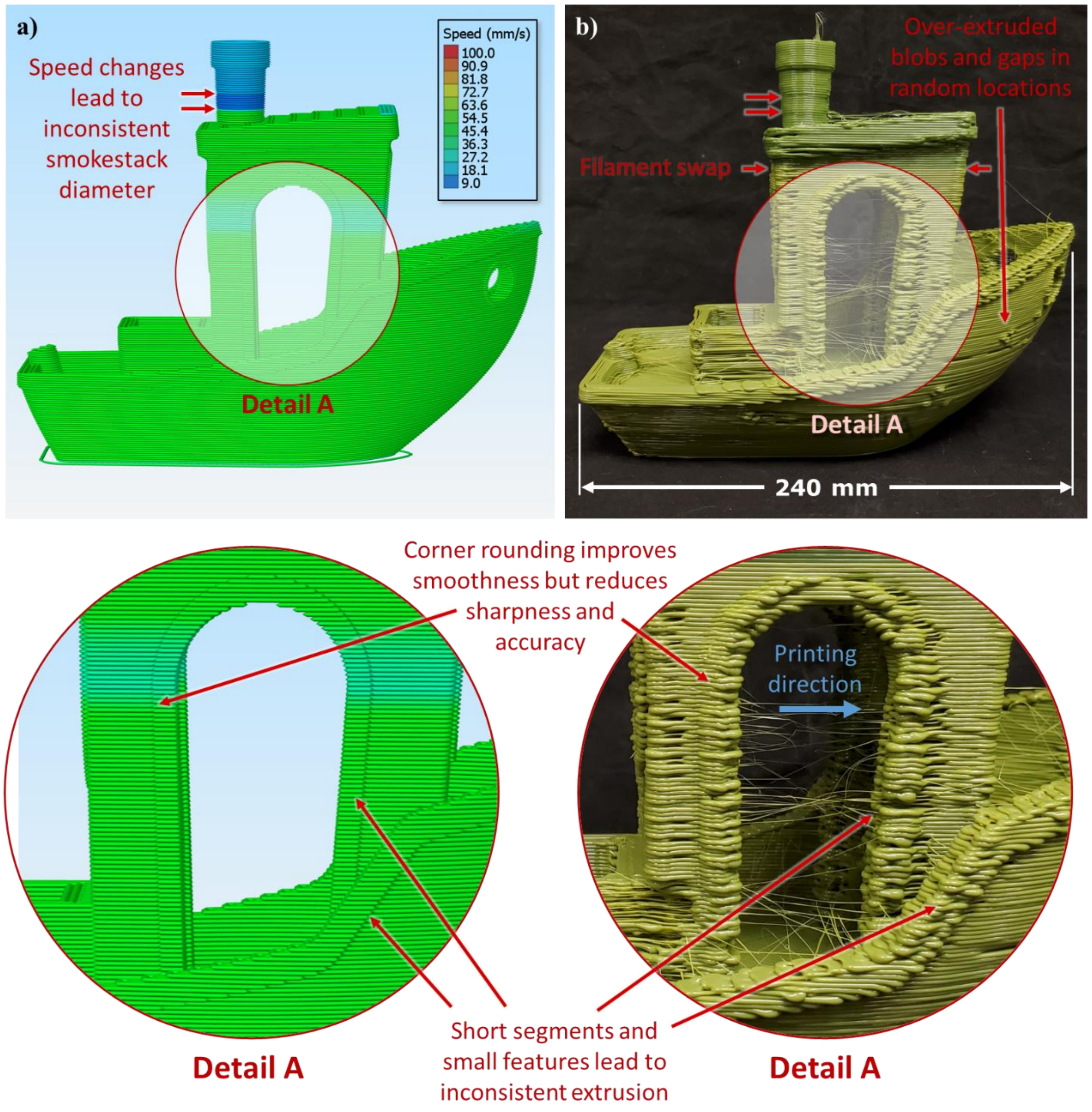

a)
Speed changes lead to inconsistent smokestack diameter
Speed (mm/s)
100.0
90.9
81.8
72.7
63.6
54.5
45.4
36.3
27.2
18.1
9.0
Detail A
b)
Over-extruded blobs and gaps in random locations
Filament swap
Detail A
240 mm
Corner rounding improves smoothness but reduces sharpness and accuracy
Printing direction
Short segments and small features lead to inconsistent extrusion
Detail A
Detail A

Figure 6-6: a) 400% scale #3DBenchy sliced in Simplify3D with detail on central portion of the bridge b) Final part produced with the Typhoon extruder on the Fanuc robot with detail on the same location

6.3 Key points

In this chapter, we used the universal printhead controller and the custom printing methodology previously developed to produce large-scale parts. We focused on printing with the Typhoon extruder on the Fanuc robot. First, we evaluated the dynamic extrusion control method introduced in Section 4.3 by printing several benchmark parts. We then used a simple analytical method to determine the Typhoon extruder’s maximum flowrate at different temperatures, as shown in Figure 6-3. These results allowed us to determine appropriate slicing parameters to print large parts as quickly as possible. We then produced two large parts: a spiral vase and a #3DBenchy. The next chapter discusses the strengths and weaknesses of the methodology we developed based on the results we obtained.

CHAPTER 7 GENERAL DISCUSSION

This chapter discusses the custom methodology we developed to implement large-scale FFF on the LM2 motion systems. More specifically, it expands on the methodology's most critical element: the link between the motion system controller and the printhead controller developed in Chapter 5. Synchronizing the two controllers depends on the serial communication protocols and dynamic extrusion speed control method implemented in Chapter 4. Having used this extrusion control method to print large parts in Chapter 6, we can now comment on a few of its strengths and weaknesses.

The dynamic extrusion control method's main strength is that it takes the motion system's real-time speed variations into account to determine material deposition rate. Although shorter moves featuring sudden direction changes still cause problems, extrusion speed now generally varies as a function of the motion system's actual speed, rather than the speed requested by the slicer. This means that acceleration and deceleration are now considered, and curved profiles print smoothly despite being approximated by short, straight-line segments in the slicer. Moreover, because the method doesn't continuously serially transmit G-code extrusion commands to the printhead controller, the Duet 3 mainboard's buffer does not fill up with instructions, resulting in better synchronization between it and the motion system controller.

Using the dynamic control method, the only time-critical commands that are sent to the printhead controller over serial are the start and stop commands at the beginning and end of print sequences. These are G-code commands which have the following form: *M594 PX*, where *X* is either 1 or 0 to indicate a start or stop instruction. This command is 7 characters long, leading to a 7ms delay in starting and stopping extrusion due to the 9600bps baud rate limitation on the Fanuc robot controller. For larger, simpler parts, these serial delays have a reasonably small impact on part quality. They mainly cause gaps at the beginning of print sequences as the start of extrusion lags the motion system velocity, and over-extrusion at the end of them for a similar reason. Assuming a constant 60mm/s print speed, these 7ms delays would cause a 0.42mm gap at the beginning of every printing sequence, and a small blob or stringing at the end of it, both of which may be acceptable on very large parts. As the part feature size goes down, the impact of these delays on part quality becomes more important.

However, these delays are dwarfed by those associated with the TCP speed analog signal and its prediction time, which caused several problems when printing large scale parts. As we saw when printing the rectangular plates and the #3DBenchy, small features and direction changes cause a mismatch between the speed reported by the analog signal and the Fanuc robot’s actual speed, leading to blobs or gaps in corners and when printing short segments. This issue seems to be related to the Fanuc robot controller’s look-ahead capabilities. As the robot executes a program, it typically reads the program lines ahead of the line currently being executed. This look-ahead capability is common on robot controllers – including the Aerotech system and the Duet 3 mainboard – because it enables useful functionality like linking moves together and reducing speed variations. As mentioned in Section 6.2.2, during sequences which contain many consecutive moves in a short amount of time, the robot will typically slow down to be able to keep reading upcoming lines adequately. Unfortunately, this impacts the TCP speed prediction: according to the Fanuc M20iB25 robot’s handling manual, “prediction becomes invalid during instances of speed and motor limit conditions” [111]. This seems to explain the issues we saw on many prints, where the extrusion speed doesn’t follow the robot’s motion during certain moves, especially when the TCP prediction time is set to higher values.

The Fanuc robot’s look-ahead functionality causing it to slow down during certain move sequences is also likely linked to the long print times we observed on the six-axis robot. We noticed this phenomenon especially when printing large parts like the vase and the #3DBenchy. During the prints, we observed that the robot’s maximum speed rarely reached the print speed that we had set in Simplify3D, which is probably the main reason for the 67% increase in real versus estimated print time. The lower printing speed also means that the Typhoon printhead’s extrusion flowrate likely didn’t reach the value we had been aiming for. Resolving the slow-down issue could lead to significant reductions in print time without exceeding the extruder’s maximum acceptable flowrate.

Up until this point, we have discussed the motion system controller’s contribution to printing defects and signal delays. However, the printhead controller also plays a role in the imperfect synchronization between extrusion and motion. The most important weakness on the printhead controller side is caused by a limitation in the Duet 3 control board’s firmware that we modified. To change the extrusion speed based on the TCP speed analog signal, we programmed the Duet mainboard to read an analog input at a set frequency, modifying the extrusion speed based on changes in the input voltage. Unfortunately, we couldn’t increase this frequency beyond 50Hz, despite the Duet 3 hardware being capable of executing similar instructions at much higher rates. This means that the extrusion speed can lag the TCP speed analog signal variations by up to 20ms, which can contribute to blobs and gaps during direction changes.

In the end, the methodology we implemented to print large scale parts on the LM2 motion systems is only as reliable as the signals sent between the motion system and printhead controllers and the delays involved in transmitting and receiving those signals. In this project, we have used benchmark print quality as a gauge for signal accuracy and timing. Future users would benefit from analysing these signals using an oscilloscope or other data acquisition method while printing various parts. Comparing them to the robot’s actual real-time speed would allow them to quantify the different types of delays which affect the system. A good place to start would be to reproduce the characterization steps undertaken by Badarinath and Prabhu in their 2021 publication [88], as discussed in Section 2.4.

There is still much work to complete to reduce signal inaccuracies and delays between the motion system and printhead controller. Unfortunately, imperfect synchronization between extrusion and motion is unavoidable when using a separate controller to manage printheads, which is a fundamental aspect of the printing methodology we developed in this project. Despite this limitation, using such a controller is also advantageous in many ways. We highlight these advantages and propose ideas to further improve synchronization in the concluding chapter.

CHAPTER 8 CONCLUSION

8.1 Summary and outcome

In this project, we developed a custom printhead controller to produce large-scale parts on industrial motion systems using different FFF-based toolheads. To complete this main goal, we divided the project into three sub-objectives. In the first sub-objective, we developed a custom FFF methodology adapted to printing on industrial motion systems. This methodology relies on an in-house extrusion control method which dynamically modulates the extrusion speed based on the motion system's real-time speed. The system drives the remaining printhead components – such as heaters and fans – using standard G-code commands sent over serial.

Having designed the appropriate methodology to print large-scale parts on the LM2 motion systems, we then began constructing the universal printhead controller. This was the project's second-sub objective. The project's focus being large-scale printing, we designed the controller around the Typhoon and Pulsar high flowrate extruders from Dyze Design. We selected a Duet 3 6HC mainboard to power the controller: this board is compatible with the Typhoon and Pulsar extruders as well as most other components normally found in FFF-based printheads.

For our third sub-objective, we used these printheads and others to produce various parts on an Aerotech gantry system and Fanuc six-axis robot. We completed the bulk of our large-scale prints with the Typhoon extruder on the Fanuc robot. Using this combination, we printed a 400mm tall, 640g spiral vase in approximately 3h, reducing print time by more than 15 hours when compared to the E3D V6 printhead, the standard in FFF. We estimate that we could print the same part in about half the time (92mins) using the production version of the Pulsar extruder, which features an even higher maximum flowrate. We also printed a 240mm long, 636g #3DBenchy in 3h20m, again saving more than 15 hours of print time versus the E3D V6.

Using a custom printhead controller allowed us to rapidly integrate a wide variety of FFF toolheads on the LM2's industrial motion systems and begin printing with them. These are the main advantages of using such a controller: printhead flexibility and ease of implementation. These two advantages distinguish this work from that of previous research: while certain groups have developed controllers to control specific FFF toolheads on industrial robots, the controller developed in this project is compatible with a wide range of printheads and multiple motion systems. Future users of both LM2 systems should be able to use this controller to quickly prototype and test new toolhead ideas. They should also be able to integrate commercial printheads on the motion systems with little mechanical, electrical, or programming work required. In a research environment, this is extremely beneficial, as it allows users to explore new concepts with fewer constraints involved in implementing them on available equipment. This flexibility comes at a cost: as seen in Chapter 6 and further discussed in Chapter 7, signal timing, accuracy, and delays between the motion system and printhead controllers have a large impact on final part quality. These signal discrepancies can lead to unacceptable printing results, counteracting the benefits such a controller provides.

Although the current printhead controller produces imperfect results, it has unlocked new possibilities for the LM2 beyond conventional FFF. At the beginning of the project, our goal was to enable large-scale printing on the LM2 industrial motion systems. This objective is now possible: we used the Typhoon extruder to create large parts in this project, and the methodology we established means future users can easily implement the production version of the Pulsar extruder to rapidly produce even bigger parts. Historically, part scale has been one of FFF's main weaknesses, preventing it from becoming a widely used method of producing end-use parts. The controller developed in this project should facilitate future research on this topic, hopefully helping bring an end to this important limitation. The printhead controller's flexibility means it can also accommodate innovative printing methods, such as continuous fiber-reinforced AM or non-planar printing. Current LM2 users have already begun leveraging the controller's capabilities to explore these ideas. If their research progresses in the same vein, this should help make FFF an industrially viable manufacturing process, with numerous advantages over conventional manufacturing methods.

8.2 Recommendations

To improve on the printing results we obtained in this project, we recommend completing the following future work:

- The analog TCP speed signal outputted by the robot controller should be characterized more thoroughly. Comparing this signal to the robotic arm's actual real-time speed as described in Chapter 7 would be highly informative: it would allow users to identify whether the robotic arm, the printhead controller or the high flowrate printheads themselves are the cause of current printing defects.

- Testing printheads other than the Typhoon extruder on the Fanuc robot (E3D V6, Pulsar, SuperVolcano, etc.) would also be helpful. It would help distinguish which printing errors are linked to the type of printhead we used versus those caused by the extrusion control method established in this project. Future work should focus on the production version of the Pulsar extruder especially, which would allow us to print even larger prints at a faster rate.
- Reproducing the dynamic extrusion speed control method on the Aerotech controller should also be completed in the near future, both for comparison purposes with the Fanuc robot and to continue large-scale printing development on that system.

Assuming TCP signal analysis concludes that the current extrusion-control method is incapable of producing better prints, there are many other ways of improving print quality:

- One option is to bring extrusion-control back to the motion system controller instead of passing through an intermediate printhead controller. Many industrial motion system controllers can drive additional motors as though they were axes on the main system: for example, adding certain hardware to the Fanuc six-axis robot controller allows it to drive a seventh axis in sync with its other joints. Connecting this axis to an extruder would eliminate acceleration mismatches between the printhead and the motion system due to delays or other signal limitations. Meanwhile, heaters, sensors and other printhead components could still be connected to the printhead controller developed in this project. This would keep most of the flexibility of the printhead controller, while ensuring time-sensitive functionality like extrusion control is subject to fewer delays.
    - One caveat with this method is that industrial motion system controllers are generally more limited in terms of motor selection. For example, the Fanuc robot controller is only capable of driving motors produced by the company, instead of standard NEMA stepper motors commonly used in FFF printheads. This means that users would need to swap the extruder motors out on commercial printheads while ensuring speed, torque and other mechanical requirements are still satisfied.

Although making this change on every new printhead is a tedious job, it may still be worthwhile if it eliminates current printing defects.

- Another option to explore to improve print quality would be to replace the straight-segment curve approximations generated by FFF slicers with proper spline-based toolpaths. On the Fanuc robot especially, these could help produce better parts while enabling higher printing speeds. To our knowledge, there are no commercial slicers capable of producing spline-based G-code; implementing this within our current methodology would require an additional post-processing step to convert the existing straight-line segments into splines that the Fanuc and Aerotech robot can execute.
    - A good intermediate step would be to implement the ArcWelderPlugin by FormerLurker within our existing methodology [112]. This plugin can be added onto many conventional slicers (including Simplify3D) and replaces combinations of straight-line g-code commands into arc-based g-code where appropriate. Although this isn’t as powerful as creating spline-based motion instructions, it should significantly reduce the number of instructions required to print geometries like the #3DBenchy, perhaps reducing the lookahead issues we spoke about in Chapter 7.

REFERENCES

[1] S. Saleh Alghamdi, S. John, N. Roy Choudhury, et N. K. Dutta, « Additive Manufacturing of Polymer Materials: Progress, Promise and Challenges », *Polymers*, vol. 13, n° 5, p. 753, févr. 2021, doi: 10.3390/polym13050753.

[2] John A. Slotwinski, « Additive Manufacturing: The Current State of the Art and Future Potential », *Johns Hopkins APL Tech. Dig.*, vol. 35, n° 4, 2021, doi: 10.1520/ISOASTM52900-15.

[3] S. D. Nath et S. Nilufar, « An Overview of Additive Manufacturing of Polymers and Associated Composites », *Polymers*, vol. 12, n° 11, p. 2719, nov. 2020, doi: 10.3390/polym12112719.

[4] N. van de Werken, H. Tekinalp, P. Khanbolouki, S. Ozcan, A. Williams, et M. Tehrani, « Additively manufactured carbon fiber-reinforced composites: State of the art and perspective », *Addit. Manuf.*, vol. 31, p. 100962, janv. 2020, doi: 10.1016/j.addma.2019.100962.

[5] ASTM Standards, « Additive manufacturing - General principles - Fundamentals and vocabulary »:, ASTM Standards. doi: 10.3403/30448424.

[6] T. Wohlers et T. Gornet, « History of additive manufacturing », p. 38, 2016.

[7] M. Rafiee, R. D. Farahani, et D. Therriault, « Multi-Material 3D and 4D Printing: A Survey », *Adv. Sci.*, vol. 7, n° 12, p. 1902307, 2020, doi: 10.1002/advs.201902307.

[8] « Guide to Part Consolidation for Additive Manufacturing ». https://www.protolabs.com/resources/guides-and-trend-reports/combining-part-assemblies-with-additive-manufacturing/ (consulté le 5 avril 2022).

[9] T. D. Ngo, A. Kashani, G. Imbalzano, K. T. Q. Nguyen, et D. Hui, « Additive manufacturing (3D printing): A review of materials, methods, applications and challenges », *Compos. Part B Eng.*, vol. 143, p. 172-196, juin 2018, doi: 10.1016/j.compositesb.2018.02.012.

[10] C. Holshouser, C. Newell, et S. Palas, « Out of Bounds Additive Manufacturing », p. 3.

[11] S. S. Crump et A. E.-P. D. Muir, « CREATING THREE-DIMENSIONAL OBJECTS », p. 15, 1992.

[12] « FDM Trademark of Stratasys, Inc. - Registration Number 1663961 - Serial Number 74133656 :: Justia Trademarks ». http://trademarks.justia.com/741/33/fdm-74133656.html (consulté le 5 avril 2022).

[13] W. L. Kempton, « A Design Sociotechnical Making of 3D Printing », dans *Additive Manufacturing*, 1re éd., S. Killi, Éd. Pan Stanford, 2017, p. 21-74. doi: 10.1201/b22510-2.

[14] K. S. Boparai, R. Singh, et J. S. Chohan, « Fused Deposition Modelling », dans *Additive Manufacturing*, 1re éd., R. Singh et J. P. Davim, Éd. First edition. | Boca Raton, FL : CRC Press/Taylor & Francis Group, 2018.: CRC Press, 2018, p. 127-186. doi: 10.1201/b22179-4.

[15] H. Gonabadi, A. Yadav, et S. J. Bull, « The effect of processing parameters on the mechanical characteristics of PLA produced by a 3D FFF printer », *Int. J. Adv. Manuf. Technol.*, vol. 111, n° 3, p. 695-709, nov. 2020, doi: 10.1007/s00170-020-06138-4.

[16] « Chaire industrielle Safran sur la fabrication additive des composites à matrice organique (FACMO) », *Directory of Experts*, 29 avril 2019. https://www.polymtl.ca/expertises/en/safran-industrial-research-chair-additive-manufacturing-organic-matrix-composites-amomc (consulté le 5 avril 2022).

[17] B. McKenna, « The Next Big Thing in 3-D Printing: Big Area Additive Manufacturing, or BAAM », *The Motley Fool*, 26 avril 2014. https://www.fool.com/investing/general/2014/04/26/the-next-big-thing-in-3-d-printing-big-area-addi-2.aspx (consulté le 6 avril 2022).

[18] B. Post *et al.*, « BIG AREA ADDITIVE MANUFACTURING APPLICATION IN WIND TURBINE MOLDS », p. 17, 2017.

[19] B. K. Post *et al.*, « Using Big Area Additive Manufacturing to directly manufacture a boat hull mould », *Virtual Phys. Prototyp.*, vol. 14, n° 2, p. 123-129, avr. 2019, doi: 10.1080/17452759.2018.1532798.

[20] K. M. M. Billah *et al.*, « Large-scale additive manufacturing of self-heating molds », *Addit. Manuf.*, vol. 47, p. 102282, nov. 2021, doi: 10.1016/j.addma.2021.102282.

[21] A. A. Hassen *et al.*, « THE DURABILITY OF LARGE-SCALE ADDITIVE MANUFACTURING COMPOSITE MOLDS », p. 11.

[22] H. B. Henderson *et al.*, « Additively Manufactured Single-Use Molds and Reusable Patterns for Large Automotive and Hydroelectric Components », *Int. J. Met.*, vol. 14, n° 2, p. 356-364, avr. 2020, doi: 10.1007/s40962-019-00379-0.

[23] C. Atkins *et al.*, « WIRE CO-EXTRUSION WITH BIG AREA ADDITIVE MANUFACTURING », p. 9.

[24] P. Chambon *et al.*, « Development of a range-extended electric vehicle powertrain for an integrated energy systems research printed utility vehicle », *Appl. Energy*, vol. 191, p. 99-110, avr. 2017, doi: 10.1016/j.apenergy.2017.01.045.

[25] L. J. Love, « UTILITY OF BIG AREA ADDITIVE MANUFACTURING (BAAM) FOR THE RAPID MANUFACTURE OF CUSTOMIZED ELECTRIC VEHICLES », p. 21.

[26] F. Talagani *et al.*, « Numerical Simulation of Big Area Additive Manufacturing (3D Printing) of a Full Size Car », *Sampe J.*, vol. 51, p. 27, juill. 2015.

[27] C. E. Duty *et al.*, « Structure and mechanical behavior of Big Area Additive Manufacturing (BAAM) materials », *Rapid Prototyp. J.*, vol. 23, n° 1, p. 181-189, 2017, doi: http://dx.doi.org/10.1108/RPJ-12-2015-0183.

[28] P. Chesser, B. K. Post, R. Lind, P. Lloyd, et L. J. Love, « CHANGING PRINT RESOLUTION ON BAAM VIA SELECTABLE NOZZLES », p. 12.

[29] A. Roschli, C. Duty, J. Lindahl, B. K. Post, P. C. Chesser, et K. T. Gaul, « INCREASING INTERLAMINAR STRENGTH IN LARGE SCALE ADDITIVE MANUFACTURING », p. 13.

[30] T. Smith *et al.*, « DUAL MATERIAL SYSTEM FOR POLYMER LARGE SCALE ADDITIVE MANUFACTURING », p. 10.

[31] J. Brackett *et al.*, « Characterizing material transitions in large-scale Additive Manufacturing », *Addit. Manuf.*, vol. 38, p. 101750, févr. 2021, doi: 10.1016/j.addma.2020.101750.

[32] « Thermwood LSAM - Large Scale Additive Manufacturing ». https://www.thermwood.com/lsam_home.htm (consulté le 6 avril 2022).

[33] S. Fathizadan, F. Ju, K. Rowe, A. Fiechter, et N. Hofmann, « A Novel Real-Time Thermal Analysis and Layer Time Control Framework for Large-Scale Additive Manufacturing », *J. Manuf. Sci. Eng.*, vol. 143, n° 1, p. 011009, janv. 2021, doi: 10.1115/1.4048045.

[34] D. Marrett, « Purdue University to Establish Thermwood LSAM Research Laboratory ». http://blog.thermwood.com/purdue-university-to-establish-thermwood-lsam-research-laboratory-blog-4-13-21 (consulté le 6 avril 2022).

[35] D. Moreno Nieto, V. Casal López, et S. I. Molina, « Large-format polymeric pellet-based additive manufacturing for the naval industry », *Addit. Manuf.*, vol. 23, p. 79-85, oct. 2018, doi: 10.1016/j.addma.2018.07.012.

[36] D. M. Sánchez, M. de la Mata, F. J. Delgado, V. Casal, et S. I. Molina, « Development of carbon fiber acrylonitrile styrene acrylate composite for large format additive manufacturing », *Mater. Des.*, vol. 191, p. 108577, juin 2020, doi: 10.1016/j.matdes.2020.108577.

[37] A. Nycz, V. Kishore, J. Lindahl, C. Duty, C. Carnal, et V. Kunc, « Controlling substrate temperature with infrared heating to improve mechanical properties of large-scale printed parts », *Addit. Manuf.*, vol. 33, p. 101068, mai 2020, doi: 10.1016/j.addma.2020.101068.

[38] V. Kishore *et al.*, « Infrared preheating to improve interlayer strength of big area additive manufacturing (BAAM) components », *Addit. Manuf.*, vol. 14, p. 7-12, mars 2017, doi: 10.1016/j.addma.2016.11.008.

[39] J. M. Pappas, A. R. Thakur, M. C. Leu, et X. Dong, « A parametric study and characterization of additively manufactured continuous carbon fiber reinforced composites for high-speed 3D printing », *Int. J. Adv. Manuf. Technol.*, vol. 113, n° 7-8, p. 2137-2151, avr. 2021, doi: 10.1007/s00170-021-06723-1.

[40] I. Wolff, « Ingersoll has Big Reputation for 'Giant Machines' ». https://www.sme.org/technologies/articles/2018/october/ingersoll-has-big-reputation-for-giant-machines/ (consulté le 6 avril 2022).

[41] Titan Robotics, « Home Page », *Titan Robotics*. https://titan3drobotics.com/ (consulté le 4 avril 2022).

[42] « Massive Dimension - Large Format 3D Printers », *Massive Dimension*. https://massivedimension.com/ (consulté le 6 avril 2022).

[43] Ø. K. Grutle, « 5-axis 3D Printer », 2015, Consulté le: 3 avril 2022. [En ligne]. Disponible à: https://www.duo.uio.no/handle/10852/47652

[44] W. Lee, C. Wei, et S.-C. Chung, « Development of a hybrid rapid prototyping system using low-cost fused deposition modeling and five-axis machining », *J. Mater. Process. Technol.*, vol. 214, n^o 11, p. 2366-2374, nov. 2014, doi: 10.1016/j.jmatprotec.2014.05.004.

[45] M. A. Isa et I. Lazoglu, « Five-axis additive manufacturing of freeform models through buildup of transition layers », *J. Manuf. Syst.*, vol. 50, p. 69-80, janv. 2019, doi: 10.1016/j.jmsy.2018.12.002.

[46] H. Peng, R. Wu, S. Marschner, et F. Guimbretière, « On-The-Fly Print: Incremental Printing While Modelling », dans *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, San Jose California USA, mai 2016, p. 887-896. doi: 10.1145/2858036.2858106.

[47] E. Barnett et C. Gosselin, « Large-scale 3D printing with a cable-suspended robot », *Addit. Manuf.*, vol. 7, p. 27-44, juill. 2015, doi: 10.1016/j.addma.2015.05.001.

[48] X. Song, X. Jin, C. Zhu, F. Cai, et W. Tian, « 3D printing and mechanical properties of glass fiber/photosensitive resin composites », *Fangzhi XuebaoJournal Text. Res.*, vol. 42, n^o 1, p. 73-77, 2021, doi: 10.13475/j.fzxb.20191105906.

[49] L. Danielsen Evjemo, S. Moe, J. T. Gravdahl, O. Roulet-Dubonnet, L. T. Gellein, et V. Br⊘tan, « Additive manufacturing by robot manipulator: An overview of the state-of-the-art and proof-of-concept results », dans *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, sept. 2017, p. 1-8. doi: 10.1109/ETFA.2017.8247617.

[50] I. F. Onstein, L. D. Evjemo, et J. T. Gravdahl, « Additive Manufacturing Path Generation for Robot Manipulators Based on CAD Models », *IFAC-Pap.*, vol. 53, n^o 2, p. 10037-10043, 2020, doi: 10.1016/j.ifacol.2020.12.2724.

[51] X. Zhang *et al.*, « Large-scale 3D printing by a team of mobile robots », *Autom. Constr.*, vol. 95, p. 98-106, nov. 2018, doi: 10.1016/j.autcon.2018.08.004.

[52] M. E. Tiryaki, X. Zhang, et Q.-C. Pham, « Printing-while-moving: a new paradigm for large-scale robotic 3D Printing », dans *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, nov. 2019, p. 2286-2291. doi: 10.1109/IROS40897.2019.8967524.

[53] H. Shen, L. Pan, et J. Qian, « Research on large-scale additive manufacturing based on multi-robot collaboration technology », *Addit. Manuf.*, vol. 30, p. 100906, déc. 2019, doi: 10.1016/j.addma.2019.100906.

[54] Y. Yao, Y. Zhang, M. Aburaia, et M. Lackner, « 3D Printing of Objects with Continuous Spatial Paths by a Multi-Axis Robotic FFF Platform », *Appl. Sci.*, vol. 11, n^o 11, p. 4825, mai 2021, doi: 10.3390/app11114825.

[55] L. Li, A. Haghighi, et Y. Yang, « A novel 6-axis hybrid additive-subtractive manufacturing process: Design and case studies », *J. Manuf. Process.*, vol. 33, p. 150-160, juin 2018, doi: 10.1016/j.jmapro.2018.05.008.

[56] B. Felbrich *et al.*, « A novel rapid additive manufacturing concept for architectural composite shell construction inspired by the shell formation in land snails », *Bioinspir. Biomim.*, vol. 13, nº 2, p. 026010, févr. 2018, doi: 10.1088/1748-3190/aaa50d.

[57] P. M. Bhatt *et al.*, « A Robotic Cell for Multi-Resolution Additive Manufacturing », dans *2019 International Conference on Robotics and Automation (ICRA)*, mai 2019, p. 2800-2807. doi: 10.1109/ICRA.2019.8793730.

[58] P. M. Bhatt, A. Kulkarni, R. K. Malhan, B. C. Shah, Y. J. Yoon, et S. K. Gupta, « Automated Planning for Robotic Multi-Resolution Additive Manufacturing », *J. Comput. Inf. Sci. Eng.*, vol. 22, nº 2, oct. 2021, doi: 10.1115/1.4052083.

[59] P. M. Bhatt, R. K. Malhan, et S. K. Gupta, « Computational Foundations for Using Three Degrees of Freedom Build Platforms to Enable Supportless Extrusion-Based Additive Manufacturing », dans *Volume 1: Additive Manufacturing; Manufacturing Equipment and Systems; Bio and Sustainable Manufacturing*, Erie, Pennsylvania, USA, juin 2019, p. V001T02A026. doi: 10.1115/MSEC2019-3024.

[60] S. Keating et N. Oxman, « Compound fabrication: A multi-functional robotic platform for digital design and fabrication », *Robot. Comput.-Integr. Manuf.*, vol. 29, nº 6, p. 439-448, déc. 2013, doi: 10.1016/j.rcim.2013.05.001.

[61] J. R. Kubalak, « Design and Realization of a 6 Degree of Freedom Robotic Extrusion Platform », 2016, p. 19.

[62] A. T. Alsharhan, T. Centea, et S. K. Gupta, « Enhancing Mechanical Properties of Thin-Walled Structures Using Non-Planar Extrusion Based Additive Manufacturing », dans *Volume 2: Additive Manufacturing; Materials*, Los Angeles, California, USA, juin 2017, p. V002T01A016. doi: 10.1115/MSEC2017-2978.

[63] G. A. Mazzei Capote, N. M. Rudolph, P. V. Osswald, et T. A. Osswald, « Failure surface development for ABS fused filament fabrication parts », *Addit. Manuf.*, vol. 28, p. 169-175, août 2019, doi: 10.1016/j.addma.2019.05.005.

[64] A. V. Shembekar, Y. J. Yoon, A. Kanyuck, et S. K. Gupta, « Generating Robot Trajectories for Conformal Three-Dimensional Printing Using Nonplanar Layers », *J. Comput. Inf. Sci. Eng.*, vol. 19, nº 3, p. 031011, sept. 2019, doi: 10.1115/1.4043013.

[65] M. Chalvin, « Layer-by-layer generation of optimized joint trajectory for multi-axis robotized additive manufacturing of parts of revolution », *Robot. Comput. Integr. Manuf.*, p. 11, 2020.

[66] G. Zhao, G. Ma, J. Feng, et W. Xiao, « Nonplanar slicing and path generation methods for robotic additive manufacturing », *Int. J. Adv. Manuf. Technol.*, vol. 96, nº 9-12, p. 3149-3159, juin 2018, doi: 10.1007/s00170-018-1772-9.

[67] C. Wu, C. Dai, G. Fang, Y.-J. Liu, et C. C. L. Wang, « RoboFDM: A robotic system for support-free fabrication using FDM », dans *2017 IEEE International Conference on Robotics and Automation (ICRA)*, mai 2017, p. 1175-1180. doi: 10.1109/ICRA.2017.7989140.

[68] I. Ishak et P. Larochelle, « Robot Arm Platform for Additive Manufacturing: 3D Lattice Structures », p. 5, 2017.

[69] I. Ishak, J. Fisher, et P. Larochelle, « Robot Arm Platform for Additive Manufacturing: Multi-Plane Printing », p. 6, 2016.

[70] B. J. Brooks, K. M. Arif, S. Dirven, et J. Potgieter, « Robot-assisted 3D printing of biopolymer thin shells », *Int. J. Adv. Manuf. Technol.*, vol. 89, n^o 1-4, p. 957-968, mars 2017, doi: 10.1007/s00170-016-9134-y.

[71] G. Q. Zhang, A. Spaak, C. Martinez, D. T. Lasko, B. Zhang, et T. A. Fuhlbrigge, « Robotic additive manufacturing process simulation - towards design and analysis with building parameter in consideration », dans *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, août 2016, p. 609-613. doi: 10.1109/COASE.2016.7743457.

[72] G. Q. Zhang, W. Mondesir, C. Martinez, X. Li, T. A. Fuhlbrigge, et H. Bheda, « Robotic additive manufacturing along curved surface — A step towards free-form fabrication », dans *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, déc. 2015, p. 721-726. doi: 10.1109/ROBIO.2015.7418854.

[73] A. V. Shembekar, Y. J. Yoon, A. Kanyuck, et S. K. Gupta, « Trajectory Planning for Conformal 3D Printing Using Non-Planar Layers », dans *Volume 1A: 38th Computers and Information in Engineering Conference*, Quebec City, Quebec, Canada, août 2018, p. V01AT02A026. doi: 10.1115/DETC2018-85975.

[74] R. J. A. Allen et R. S. Trask, « An experimental demonstration of effective Curved Layer Fused Filament Fabrication utilising a parallel deposition robot », *Addit. Manuf.*, vol. 8, p. 78-87, oct. 2015, doi: 10.1016/j.addma.2015.09.001.

[75] I. Bin Ishak, J. Fisher, et P. Larochelle, « Robot Arm Platform for Additive Manufacturing Using Multi-Plane Toolpaths », dans *Volume 5A: 40th Mechanisms and Robotics Conference*, Charlotte, North Carolina, USA, août 2016, p. V05AT07A063. doi: 10.1115/DETC2016-59438.

[76] T. Felsch, F. Silze, et M. Schnick, « Process Control for Robot Based Additive Manufacturing », dans *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 10-13 Sept. 2019*, Piscataway, NJ, USA, 2019, p. 1489-92. doi: 10.1109/ETFA.2019.8869530.

[77] CEAD, « Solutions for large scale industrial 3D printing - CEAD Group », *CEAD | Composite Additive Manufacturing*. https://ceadgroup.com/solutions/ (consulté le 31 décembre 2021).

[78] Ai Build, « Ai Build | Enterprise software that unlocks the true potential of additive manufacturing », 22 mars 2022. https://ai-build.com (consulté le 31 mars 2022).

[79] ABB, « 3D Printing PowerPac », *Robotics*. https://new.abb.com/products/robotics/application-software/3d-printing-powerpac (consulté le 4 avril 2022).

[80] Yaskawa, « Additive Manufacturing - Yaskawa ». https://www.yaskawa.com/industries/additive-manufacturing (consulté le 7 avril 2022).

[81] KUKA, « Automation and additive manufacturing », *KUKA AG*. https://www.kuka.com/en-ca/products/process-technologies/3d-printing (consulté le 4 avril 2022).

[82] J. Duro-Royo, L. Mogas-Soldevila, et N. Oxman, « Flow-based fabrication: An integrated computational workflow for design and digital additive manufacturing of multifunctional heterogeneously structured objects », *Comput.-Aided Des.*, vol. 69, p. 143-154, déc. 2015, doi: 10.1016/j.cad.2015.05.005.

[83] C. Gosselin, R. Duballet, Ph. Roux, N. Gaudillière, J. Dirrenberger, et Ph. Morel, « Large-scale 3D printing of ultra-high performance concrete – a new processing route for architects and builders », *Mater. Des.*, vol. 100, p. 102-109, juin 2016, doi: 10.1016/j.matdes.2016.03.097.

[84] J. Werner, M. Aburaia, A. Raschendorfer, et M. Lackner, « MeshSlicer: A 3D-Printing software for printing 3D-models with a 6-axis industrial robot », *Procedia CIRP*, vol. 99, p. 110-115, 2021, doi: 10.1016/j.procir.2021.03.018.

[85] A. Dine et G.-C. Vosniakos, « On the development of a robot-operated 3D-printer », *Procedia Manuf.*, vol. 17, p. 6-13, 2018, doi: 10.1016/j.promfg.2018.10.004.

[86] I. Ishak, J. Fisher, et P. Larochelle, « Robot Arm Platform for Rapid Prototyping: Concept », 2015.

[87] P. Urhal, A. Weightman, C. Diver, et P. Bartolo, « Robot assisted additive manufacturing: a review », *Robot. Comput.-Integr. Manuf.*, vol. 59, p. 335-45, oct. 2019, doi: 10.1016/j.rcim.2019.05.005.

[88] R. Badarinath et V. Prabhu, « Integration and evaluation of robotic fused filament fabrication system », *Addit. Manuf.*, vol. 41, p. 101951, mai 2021, doi: 10.1016/j.addma.2021.101951.

[89] « Documentation - ROS Wiki ». https://wiki.ros.org/ (consulté le 11 avril 2022).

[90] « ROS: Home ». https://www.ros.org/ (consulté le 11 avril 2022).

[91] « What is ROS? », *Ubuntu*. https://ubuntu.com/robotics/what-is-ros (consulté le 11 avril 2022).

[92] Thingiverse.com, « Thingiverse - Digital Designs for Physical Objects ». https://www.thingiverse.com/ (consulté le 21 mars 2022).

[93] « 3D models database », *PrusaPrinters*. https://www.prusaprinters.org/prints (consulté le 21 mars 2022).

[94] « Ultimaker Cura: Powerful, easy-to-use 3D printing software », *ultimaker.com*. https://ultimaker.com/software/ultimaker-cura (consulté le 21 mars 2022).

[95] « PrusaSlicer | Original Prusa 3D printers directly from Josef Prusa ». https://www.prusa3d.com/page/prusaslicer_424/ (consulté le 21 mars 2022).

[96] « Slic3r - Open source 3D printing toolbox ». https://slic3r.org/ (consulté le 21 mars 2022).

[97] « Professional 3D Printing Software | Simplify3D ». https://www.simplify3d.com/ (consulté le 21 mars 2022).

[98] « V6 All-Metal HotEnd », *E3D Online*. https://e3d-online.com/products/v6-all-metal-hotend (consulté le 5 avril 2022).

[99] « Bondtech Mini Geared (BMG™) Extruder for groove mount E3D Hotends », *Bondtech*. https://www.bondtech.se/product/bmg-extruder/ (consulté le 5 avril 2022).

[100] « Measure and calibrate », *#3DBenchy*, 7 avril 2015. https://www.3dbenchy.com/dimensions/ (consulté le 24 mars 2022).

[101] « Features », *#3DBenchy*, 7 avril 2015. https://www.3dbenchy.com/features/ (consulté le 24 mars 2022).

[102] J.-F. Chauvette, D. Brzeski, I. L. Hia, R. D. Farahani, N. Piccirelli, et D. Therriault, « High-speed multinozzle additive manufacturing and extrusion modeling of large-scale microscaffold networks », *Addit. Manuf.*, vol. 47, p. 102294, nov. 2021, doi: 10.1016/j.addma.2021.102294.

[103] « Duet3D Forum », *Duet3D*. https://forum.duet3d.com (consulté le 22 mars 2022).

[104] « Duet3D Documentation », *Duet3D Documentation*. https://docs.duet3d.com/ (consulté le 22 mars 2022).

[105] « Duet3D Limited », *GitHub*. https://github.com/Duet3D (consulté le 22 mars 2022).

[106] *lm2-poly/RepRapFirmware*. Lm^2 - Polytechnique Montreal, 2022. Consulté le: 12 avril 2022. [En ligne]. Disponible à: https://github.com/lm2-poly/RepRapFirmware

[107] « Typhoon™ 2.85mm Filament Extruder | DYZE DESIGN Docs ». https://docs.dyzedesign.com/typhoon.html#printing-speed (consulté le 29 mars 2022).

[108] « Max volumetric speed », *Prusa Knowledgebase*. https://help.prusa3d.com/en/article/max-volumetric-speed_127176/ (consulté le 29 mars 2022).

[109] « Pulsar™ Pellet Extruder | DYZE DESIGN Docs ». https://docs.dyzedesign.com/pulsar.html#slicer-guidelines (consulté le 25 avril 2022).

[110] L. Ford, « Ingeo Biopolymer 3D850 Technical Data Sheet », p. 5.

[111] FANUC America Corporation, « FANUC AMERICA CORPORATION SYSTEM R-30i B PLUS/R-30iB MATE PLUS/R-30iB COMPACT PLUS HANDLINGTOOL SETUP AND OPERATIONS MANUAL », p. 2516.

[112] FormerLurker, *FormerLurker/ArcWelderPlugin*. 2022. Consulté le: 29 avril 2022. [En ligne]. Disponible à: https://github.com/FormerLurker/ArcWelderPlugin

[113] « Position Synchronized Output (PSO) | Aerotech », *Aerotech US*, 27 octobre 2020. https://www.aerotech.com/position-synchronized-output-pso-coordinate-part-position-with-process-control/ (consulté le 23 mars 2022).

[114] « Duet 3 Mainboard 6HC », *Duet3D Documentation*. https://docs.duet3d.com/Duet3D_hardware/Duet_3_family/Duet_3_Mainboard_6HC_Hardware_Overview (consulté le 23 mars 2022).

[115] Thingiverse.com, « Spiral vase by JJ76 ». https://www.thingiverse.com/thing:2795194 (consulté le 25 mars 2022).

APPENDIX A ELECTRIC COMPONENTS TYPICALLY FOUND IN FFF PRINTHEADS

Section 3.2.3 overviews the main electric components typically found in FFF printheads. This appendix provides more details on these components and how they interact with an electronic control board.

Extruder motor

Consumer extruders for FFF printers typically use bipolar stepper motors to drive the base thermoplastic material. Stepper motors are available in standard frame sizes ranging from small, 20mm-diameter NEMA 8 motors up to 102mm-diameter NEMA 42 motors, with NEMA 17 and NEMA 23 motors most used in FFF extruders and printers. Stepper motor resolution, torque and current ratings vary within a frame size, with larger motors typically producing more torque and requiring higher currents.

Unlike servomotors, which spin continuously and typically use encoders to monitor their position and function as a closed-loop system, stepper motors have no such feedback device; instead, they move incrementally from one rotational position to the next, called *steps*. To control a stepper motor, it must be connected to a stepper driver. The driver regulates the current through the motor windings based on input signals from a control board, making the motor advance a specified number of steps by energizing its windings in the correct order. Stepper drivers have maximum voltage and current specifications which must be sufficient for the stepper motor they are paired with based on application speed and torque requirements. FFF printers typically drive extruder stepper motors at 12 or 24V (higher voltages are sometimes used), with currents ranging between 500mA and 3A.

Hot end heater

Most FFF hot ends use one form or another of resistive heating to melt the incoming thermoplastic for deposition. Smaller hot ends typically use cartridge heaters inserted into a conductive material like aluminum or copper which houses the output nozzle. This arrangement appears on the ubiquitous E3D V6, for example, as shown in Figure 3-3. Meanwhile, larger hot ends like the Dyze

Design Pulsar or Typhoon extruders use multiple coil heaters to heat cylindrical heater blocks, achieving similar results for higher-flowrate printing.

Electrical specifications for resistive heaters vary significantly from one printhead to the next. Certain heaters like those found in the E3D V6 run on direct current and only require a few watts of power; these most commonly operate at 12 or 24VDC and can be connected to standard analog output on a control board, or to a digital output capable of pulse-width modulation (PWM). Other higher-power heaters like those in the Pulsar pellet extruder run on mains voltage; controlling them using standard I/O on a control board requires relays, as discussed in Section 5.2.2.3.

Hot end temperature sensor

The two most common temperature sensors in consumer FFF printers are thermistors and platinum resistance temperature detectors (RTDs), while thermocouples are also occasionally used. Thermistors and RTDs both exhibit a change in resistance in response to temperature variations, while thermocouples produce a temperature-dependent voltage difference. Obtaining usable temperature measurements from any of these three sensors requires sensor-specific circuitry and conversion tables or equations.

Thermistors are typically arranged in a voltage divider with a known, fixed resistance. The divider’s output voltage is read by an analog input on a control board, used to calculate the thermistor’s resistance value and then converted to temperature using the well-known Steinhart-Hart equation: $\frac{1}{T} = a + b\ln(R) + c\ln^3(R)$, where R is the thermistor’s resistance at a specific temperature T, and a, b and c are the Steinhart-Hart coefficients, usually provided by the manufacturer. Because thermistor resistance varies significantly with temperature within its usable range, no amplification circuitry is required.

RTDs function similarly to thermistors in that their resistance varies with temperature; however, because of platinum’s near-linear resistance response to temperature, they are typically more accurate over a wider range of temperatures and require fewer data corrections to interpret temperature. This makes them particularly useful in printheads used to print both consumer thermoplastics like PLA at reasonably low temperatures and high-temperature, engineering thermoplastics like PEEK or PEI. Compared to thermistors, RTDs typically produce a much shallower resistance-temperature curve; they must therefore pass through a signal conditioning circuit before being measured by a standard control board analog input.

Although thermocouples are used less commonly in consumer FFF printers and printheads, they frequently appear in other applications. Unlike thermistors and RTDs, thermocouples are active sensors and directly produce a readable, temperature-dependent output voltage. However, this voltage is usually very small and is affected by ambient temperature: it must therefore pass through a compensation circuit before being connected to a standard analog input on a control board.

Fans

Many FFF printheads feature one or several fans for part-cooling and to prevent the cold end from getting hot, ensuring the filament stays solid before reaching the melt zone. Hot end fans are usually thermostatically controlled, while G-code commands control part-cooling fans, turning them on or off and changing their speed. This functionality can be useful when printing materials like PLA, which benefit from additional cooling on most layers, but tend to pull away from the print surface if the first layers are cooled too aggressively.

Printhead fans typically operate at 12 or 24VDC and only require a few milliamps of current. Fans are available in two, three, and four-wire configurations. Two-wire fans have a power and ground wire; applying a PWM waveform to the power wire allows a control board to vary their speed. Three-wire fans add a tachometer wire to be able to read the fan’s speed during operation, which can make it easier to identify fan failures, for example. Four-wire fans add a dedicated PWM input, with the full voltage level (12 or 24VDC depending on the fan) always applied to the power wire.

Other printhead components

FFF printers often use proximity sensors (inductive, capacitive, etc.) to properly calibrate the distance between the nozzle and the printing surface on the first layer, sometimes even compensating for distortion in the print surface or motion system after probing the surface in several locations. Other components occasionally installed on printheads include small servomotors, microswitches, actuators, etc. These typically require very little power and can be connected to standard analog and digital I/O.

APPENDIX B ESTABLISHING SERIAL COMMUNICATION ON THE AEROTECH GANTRY SYSTEM

The PSO connectors on the Aerotech system – certain of which operate at 5V TTL levels – normally control optocouplers or other devices which activate based on TTL signals from the controller. The Aerotech system can drive these devices based on the current position of its stages, producing accurate, position-dependent functionality that isn't affected by toolhead velocity or acceleration. This capability is particularly useful when driving high-pulse laser cutters or engravers which should be turned on and off based on the tool's position over the work surface. Although these outputs aren't intended for serial communication, we found a way to adapt them for this purpose by using an AeroBasic software trick that M. Bherer-Constant came up with.

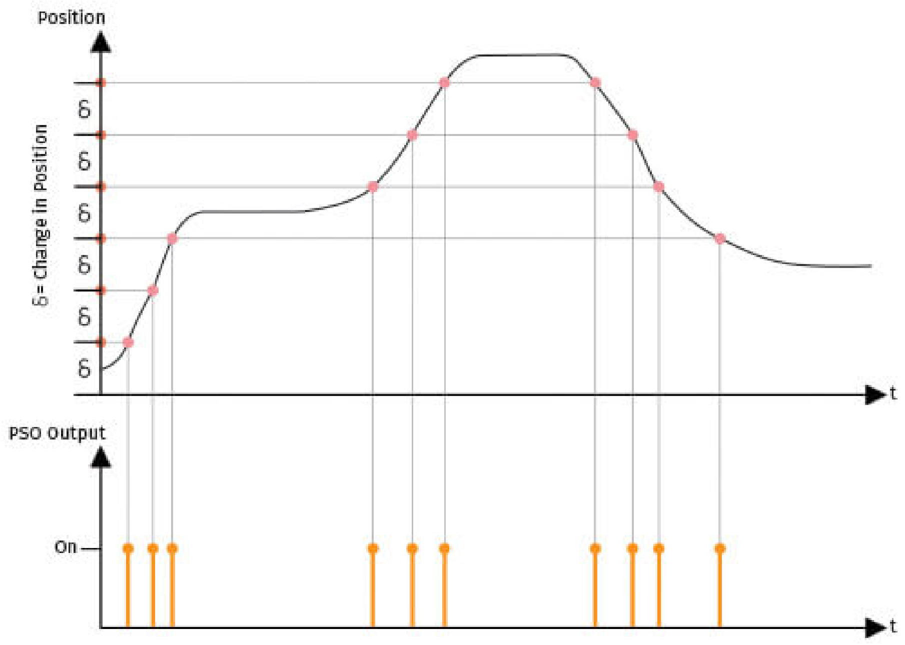


Figure B-1: PSO pulse output triggered by change in position [113]

The PSO connections on the Aerotech system have the following intended functionality, as shown in Figure B-: each time a specified position-based requirement is satisfied, such as when a selected axis' position changes by a predefined interval δ, a *firing event* occurs, which corresponds to each orange pulse in Figure B-. Users create a predefined array of pulses that they wish to output at specific locations during program execution. Aerotech calls this array the *PSO output pulse train*, which occurs at each firing event. This means that each orange pulse in the above figure is actually a rapid succession of predefined pulses. Firing events can each trigger identical output pulse trains or different ones based on how users configure them. Although this functionality is unrelated to serial communication, we noticed a few similarities that we thought we could exploit to use it to send G-code commands: first, the concept of a predefined output pulse train is similar to that of signal patterns representing data in serial communication; second, the PSO outputs can operate at high frequencies similar to those encountered in serial communication; and finally, users can customize the pulse patterns and send different ones at different points during a print. We concluded that the PSO outputs could thus allow us to encode specific G-code commands we wanted to send.

However, we had to overcome one important restriction: the pulse patterns we wanted to send weren't position dependent, but instead had to occur at a constant, specific baud rate. Even occasional, minor deviations from this baud rate would turn a G-code command sent to the Duet mainboard into a string of unrecognizable characters at best, causing nothing to happen, or worse, into the wrong commands. To fix this issue, M. Bherer-Constant suggested using the concept of virtual axes in Aerobasic. These axes respond to commands in the same way as real, hardware axes on the machine and are usually used to simulate and debug complex program functionality before implementing it on the actual system. We used this notion quite differently: we created a virtual axis in Aerobasic that moved at a constant speed during program execution. This way, when we wanted to send a G-code command to the printhead controller, we could trigger a firing event based on that axis's current position, with the pulse frequency determined by the virtual axis' speed. By tuning this speed to match the Duet 3 control board's baud rate, we could thus transmit commands serially.

There are several limitations to using the PSO outputs for serial communication, many of which we solved through extensive debugging:

1. Each ASCII character in every G-code command must be encoded into the correct pulse pattern. Unlike UARTs, which feature this functionality natively, on the Aerotech, this adds additional processing time. Each character must first be converted into the appropriate binary number which must then be transformed into a sequence of bits that is coherent with the Aerotech's implementation of PSO output pulse trains. We chose to implement this conversion step before starting the print: the Aerotech RoboDK post-processor developed

by M. Bherer-Constant converts all G-code commands into the appropriate output pulse trains, which it places in a dedicated file. When launching a print file, this file is compiled simultaneously and called upon each time a G-code command is sent. Doing so keeps the main print file's length manageable and reduces file loading times at the beginning of a print.

2. Although this method permits the Aerotech to send G-code commands to the printhead controller, it's only a one-way connection: it doesn't allow it to receive any responses from the latter. Certain G-code commands – like setting a heater to a certain temperature – normally wait for a response indicating that the requested task has been completed before continuing program execution. This one-way link is a minor inconvenience that can always be addressed using other I/O. For example, we could designate a specific digital output on the printhead controller to briefly turn on each time it completes a G-code command which requires a response. Although we haven't yet implemented this functionality, doing so would be trivial.

3. Finally, the TTL-level outputs on the Aerotech controller aren't directly compatible with the UART on the Duet 3 mainboard, for two main reasons. First, while all Duet 3 inputs are 32V-tolerant – meaning input voltages up to that level will not damage the board – the board's UART connector expects a 3.3V TTL signal whereas the PSO connectors output 5V signals. Second, the PSO TTL outputs on the Aerotech controller remain at 0V by default, moving to 5V when an output pulse is commanded. This is the exact opposite to traditional UART TTL serial as implemented on the Duet control board, which remains high when idle, with a low start bit indicating the beginning of the packet. We solved each of these issues respectively by installing both a logic-level converter and an IC inverter in the universal printhead controller before the UART connector on the Duet 3 mainboard, as detailed in Section 5.2.2.7.

APPENDIX C DUET 3 MAINBOARD 6HC MAIN SPECIFICATIONS

The following table outlines the Duet 3 Mainboard 6HC's main specifications

Table C-1: Duet 3 Mainboard 6HC main specification overview (information from [114])

Category	Component
Power distribution	• 12-32V input voltage • Onboard 3.3V, 5V and 12V produced for I/O and fans if desired
Processing	• 32-bit, 300MHz ARM Cortex M7 processor
Stepper motor control	• 6 Trinamic 2160 stepper drivers running at up to 4.45A RMS, 6.3A peak current
Heater and fan control	• 10 PWM outputs with different current capacities to control heaters and fans, including: o One high-current output (up to 18A) intended for a high power, direct current heater o Three medium-current outputs (up to 6A) intended for typical hot end heaters o Three 4-pin, low-current outputs (up to 2.5A) intended for speed-controllable fans o Three 2-pin, low-current outputs (up to 2.5A) intended for 2-wire fans
Temperature sensors	• 4 temperature sensor inputs for thermistors and PT1000 RTDs • Expansion header for PT100 or thermocouple daughterboards (up to 4 sensors)
I/O	• 9 general I/O headers to implement other functionality such as: o Connecting to other accessories like microswitches, probes, filament monitors or other sensors o Sending or receiving analog and digital signals from other controllers o Exchanging data via 3.3V UART
Connectivity	• One CAN-FD bus to connect to other Duet 3 expansion and tool boards • One Ethernet port for networking, including connecting to the board over the internet to control it or modify configuration files • One microUSB port to communicate with a computer

APPENDIX D ENCLOSURE PANEL MACHINING

The front and back enclosure panels are removable, making it easy to install them on a CNC router. To generate machining toolpaths for all the openings, we used Fusion360 software, a CAD/CAM package by Autodesk that is free for education and hobbyist use. Figure D-1a shows the simulated toolpaths for the back panel, while Figure D-1b shows the real panel being machined. Table D-1 lists the machining parameters we used. We generated two types of toolpaths for the back panel: one contour toolpath to cut out all full-depth openings and one adaptive clearing toolpath to locally reduce the panel thickness for the D-sub connectors, which required a thinner mounting panel than the 1.9mm enclosure thickness. These toolpaths are illustrated in Figure D-1c. On the front panel, we used the same kind of contour toolpath for the LCD and power switch openings, and a circular toolpath to machine the LCD panel's mounting holes and speaker opening. We completed all machining on a Shapeoko 3 XL CNC router. We first machined plywood prototypes to validate the toolpaths and test out connector locations and fit, then cut the openings in the final aluminum front and back panels. Total machining time for both panels was approximately 38 minutes, excluding setup time.

Table D-1: Main machining parameters used to machine the front and back aluminum enclosure panels for the universal printhead controller

Parameter	Value
Endmill type	Flat
Endmill diameter	1/8 inch (3.125mm)
Spindle speed	30 000 rpm
Cutting feed rate	18 in/min (457mm/min)
Plunging feed rate	9 in/min (229mm/min)
Maximum stepdown for contour and circular toolpaths	0.015in (0.4mm)
Maximum stepdown for adaptive clearing	0.030in (0.8mm)

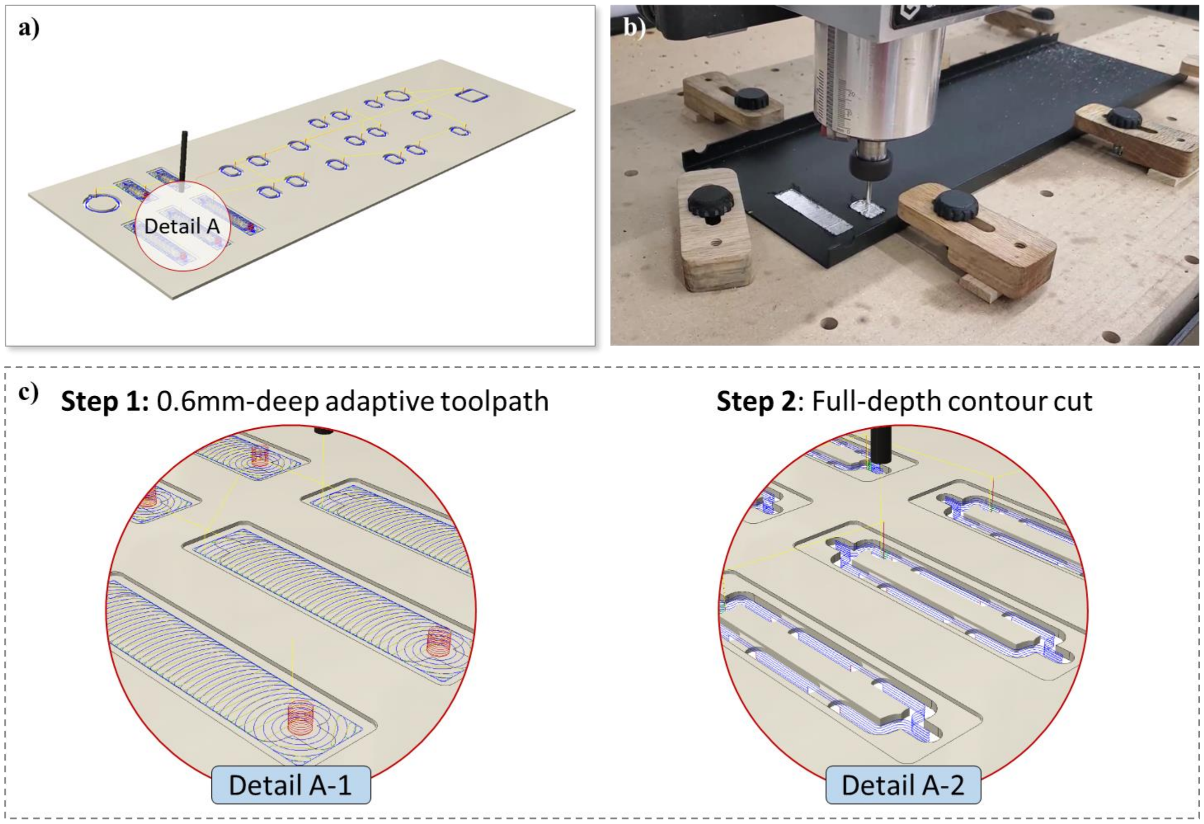

a)
Detail A
b)
c)
Step 1: 0.6mm-deep adaptive toolpath
Step 2: Full-depth contour cut
Detail A-1
Detail A-2

Figure D-1: a) Fusion360 simulated toolpaths for machining the enclosure back panel b) Machining the back panel on the Shapeoko 3 XL CNC router c) Adaptive and contour toolpath simulations in Fusion3

APPENDIX E STATIC EXTRUSION CONTROL METHOD

How it works

For our first test prints on the LM2 motion systems, we transmitted extrusion commands to the printhead controller using a temporary, static extrusion control method which we knew was suboptimal. The method – which is illustrated in Figure E-1 – worked as follows:

1. The print file outputted by the slicer contains many G-code commands which involve both printhead motion and extrusion. Going back to the example presented in Section 3.2.1.3, suppose one line of G-code in the file is *G1 X10 Y15 E2 F2000*. Assuming metric units are used, this tells a printer to move to the coordinate $(X, Y) = (10mm, 15mm)$ at 2000 mm/min while maintaining its current height in Z and extruding 2mm of filament. Conventional printers follow this instruction directly, synchronising their axes and extruders so that they all start and end at the same time. RoboDK takes this line of G-code and separates it into three consecutive instructions: an extruder call with a 2mm input parameter, a speed-setting instruction, and a move instruction to the requested coordinate.
2. The post-processor deals with each instruction sequentially. When it sees an extruder call, it sets the amount of material to extrude during the next move to its input value, 2mm in this case. Then, it does the same kind of thing upon seeing the speed-setting instruction, adjusting the speed for the next move to the instruction's input value, 2000mm/min in this case. Finally, when the post-processor reaches the move instruction, it determines what to do based on the value of the extruder and speed variables set previously. Our suboptimal static extrusion control method appears begins here and is programmed as follows:
    a. First, the post-processor calculates the average extrusion speed based on the move's requested speed using the following formula:

    $$F_{extrusion} = \frac{E}{t} = E \times \frac{F}{d} = F_{print} \times \frac{E}{\sqrt{X^2 + Y^2}} = 222mm/min$$

    Where t is the move duration, which the post-processor obtains using the distance travelled $d = \sqrt{X^2 + Y^2}$ and requested printing speed F_{print}.

b. Then, it adds an extrusion instruction to the output print file that will be launched on the motion system controller. The instruction, whose exact form depends on the motion system, essentially tells it to serially transmit the following G-command to the printhead controller: *G1 E2.00 F222*. This instruction tells the controller to extrude the requested amount at the average speed determined in the previous step.

c. Finally, the post-processor adds a move instruction to the output print file telling the motion system to move to the requested coordinates.

Separating motion and extrusion commands as demonstrated in Figure E-1 means a single G-code command in the print file generated by the slicer becomes two consecutive commands in the final output file produced by RoboDK: one serial communication command telling the printhead to extrude, followed by a motion command.

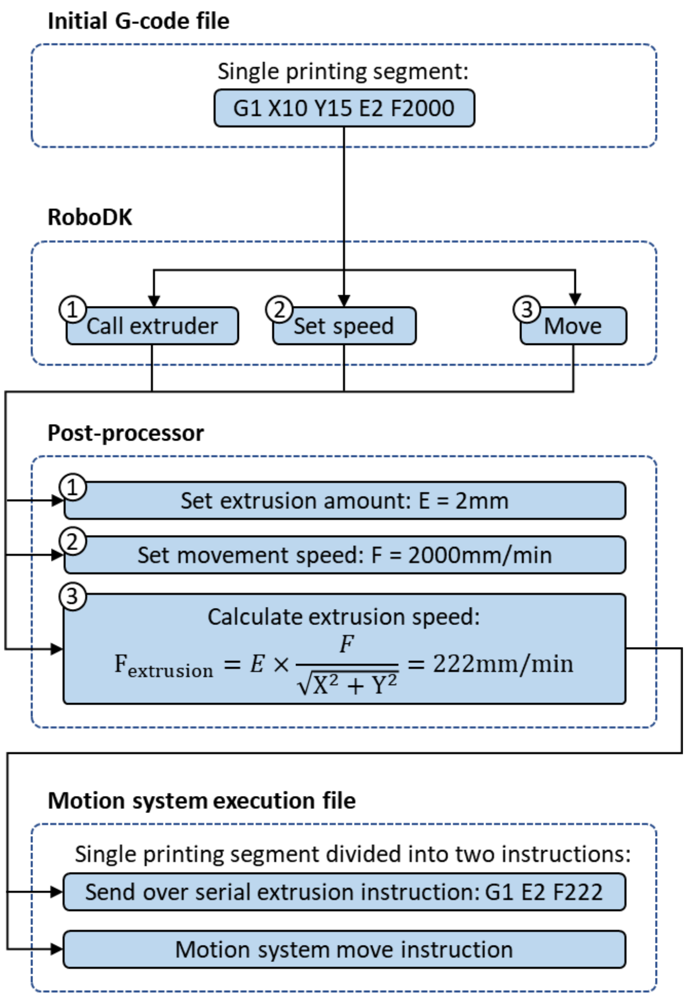

Initial G-code file
Single printing segment:
G1 X10 Y15 E2 F2000
RoboDK
1 Call extruder
2 Set speed
3 Move
Post-processor
1 Set extrusion amount: E = 2mm
2 Set movement speed: F = 2000mm/min
3 Calculate extrusion speed:
$F_{extrusion} = E \times \frac{F}{\sqrt{X^2 + Y^2}} = 222\text{mm/min}$
Motion system execution file
Single printing segment divided into two instructions:
Send over serial extrusion instruction: G1 E2 F222
Motion system move instruction

Figure E-1: Flow of data in the static extrusion control method

Limitations

The temporary static extrusion control method we created has only one main advantage: it was very simple to implement on both LM2 motion systems, allowing us to quickly begin testing printheads, as demonstrated in Appendix F and Appendix G. The method is flawed for several reasons.

First, it doesn’t consider the motion system nor the extruder’s acceleration and deceleration, only their average speed. Failing to consider acceleration leads to inconsistencies in extrusion whenever the extruder and motion system’s speeds are mismatched, which happens at the beginning and end of all segments. We observed the effect of speed mismatches on all our test parts, with blobs or under-extruded regions appearing wherever there were sudden direction changes. Generally, extruders have much lower inertia than motion system axes, which allows users to set much higher maximum allowable acceleration values for them. Therefore, as the motion system gradually accelerates, the extruder – which reaches its top speed nearly instantaneously – tends to over-extrude, creating a blob. During the constant-speed portion of a segment, the extruder and motion system move harmoniously. Then, during the deceleration portion of the move, the extruder finishes depositing the requested material quantity before the motion system has finished moving, which leads to a gap in the part. We observed larger defects of this type on the Fanuc robot because it usually accelerates more slowly than the Aerotech system.

Our static extrusion control method can cause even bigger problems when printing curved profiles, mainly because slicers typically express curves as combinations of short, straight-line segments. While the motion system controller links these segments together to produce smooth motion, it sends discrete extrusion commands to the printhead controller, causing the extruder to accelerate and decelerate between each command. This leads to jerky, uneven extrusion in curves, which we saw on the Fanuc robot and on our first prints with the Aerotech system.

However, we also saw that we managed to somewhat resolve the uneven extrusion in curves on the Aerotech system for certain prints. We believe that the issue mostly disappeared due to extrusion commands being sent to the extruder at a slightly faster pace than it can complete them. Like most motion systems, the Duet 3 mainboard uses motion ‘look-ahead’ to link upcoming moves together: if it receives new commands before finishing the current one, it can link them together without stopping, producing smooth extrusion. Linking moves together coincidentally leads to smoother curves, but it also has its drawbacks. Because new moves are sent to the controller before the previous ones are completed, they accumulate in the printhead controller's buffer. This causes extrusion commands to be executed later than they should be, potentially causing over-extrusion at the end of segments and increased stringing. Although it's difficult for us to say for certain that moves accumulating in the buffer is what caused curves to sometimes print more smoothly on the Aerotech system, it seems consistent with the types of printing defects we observed during initial testing.

Finally, another drawback to our initial extrusion-control method is that single G-code lines are decomposed into two separate instructions by RoboDK so that they may be sent to the appropriate controllers. These instructions are executed one after another during a print, inevitably leading to delays between motion and extrusion. As mentioned in Section 4.2.2, the serial port on the Fanuc controller's baud rate is limited to 9600bps, or 960 characters per second. As we saw before, a typical G-code extrusion command sent to the printhead controller using this method looks like this: *G1 E2.00 F222*, with the exact extrusion amount and speed varying from one segment to the next. This type of command needs to be sent for every segment in a print. At sixteen characters in length, this means a delay of approximately 17ms before extrusion. This doesn't consider the time it takes for the Fanuc robot to run the Karel code required to send the command and the time the Duet mainboard takes to begin extrusion. In theory, these two delays should be insignificant in comparison to the serial transmission time, but further testing is required to evaluate how large they are in practice. Shortening the number of characters in a command to reduce transmission time is possible, but it means compromising extrusion speed or volumetric accuracy.

To put these kinds of delays into context, if a curved portion of a print is approximated by 1mm-long straight segments and the default printing speed is 100mm/s, each segment takes 10ms to print. This scenario is far from unreasonable; in fact, shorter segment sizes frequently appear in FFF printing. This makes it obvious that sending a G-code command with a 17ms delay per segment will always lead to unacceptable print quality and instructions accumulating in the printhead controller's buffer.

APPENDIX F INITIAL PRINTING TESTS ON THE AEROTECH GANTRY SYSTEM

This appendix describes the initial troubleshooting we completed on the Aerotech gantry system using the static extrusion control method presented in Appendix E. We later developed a dynamic extrusion speed-control method to produce higher quality prints, as described in Section 4.3.

Standard Dyze Design printheads (DyzEnd/DyzeXtruder Pro)

We began by testing the standard Dyze Design printheads that we already had been using for several projects, including on the Aerotech and Fanuc motion systems using Mëkanic's FFF controllers. Our objective was to validate that the printhead controller was functional by pairing it with a simple printhead commonly used in the LM2. We used 3DTrip PLA and a 0.4mm nozzle for all printing tests. We first printed basic square prisms to improve print settings and debug various implementation issues. We then decided to print the "LM2" letters as two layers to see how the printhead controller would respond to slightly more complex geometries, including curves and lines of varying lengths. Figure F-1 shows the results, which we printed using a 0.2mm layer height, 60mm/s printing speed and 190ºC nozzle temperature. For each letter, we printed two perimeter outlines before printing the infill at ±45º.

Although the LM2 letters are easily distinguishable, they contain many defects. The most obvious of these are: under-extrusion in most but not all parts of the print; over-extruded blobs in the corners, where the printhead decelerates and accelerates to change direction; and regular blobs at seemingly fixed intervals in certain curved portions of the print. Despite changing different printing settings in the slicer and on the Aerotech system, we couldn't make these defects disappear completely.

We managed to mostly resolve the defects in curves by discovering certain settings on the Aerotech system: we turned on corner rounding, which rounds off sharp corners by a predefined radius, and velocity profiling, which tries to maintain motion velocity between different segments instead of stopping at every point. It does this by predicting upcoming moves and linking them together rather than printing them discretely, which is particularly important because slicers approximate curves by using short, straight-line segments. These two settings led to much smoother motion and improved the corners and curved areas of the print significantly at a slight cost to geometrical accuracy.

Modifying the extrusion multiplier did increase the amount of extruded material overall but led to certain areas being over-extruded while leaving others insufficiently extruded. We concluded that most of the printing defects we had observed were most likely linked to the way in which we were sending extrusion commands to the printhead controller, as addressed in the next chapter. On the positive side, the controller did seem capable of driving the Dyze Design printheads and responded to changes in print settings the way we would expect it to.

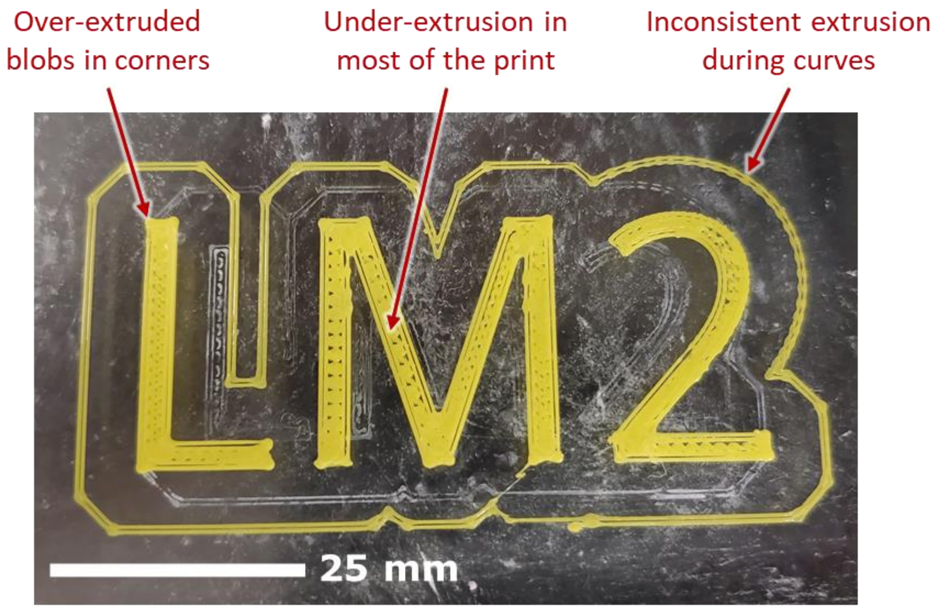

Over-extruded blobs in corners
Under-extrusion in most of the print
Inconsistent extrusion during curves
25 mm

Figure F-1: ‘LM2’ letters printed on the Aerotech system using a standard Dyze Design printhead with main defects highlighted

Typhoon extruder

After mounting the Typhoon extruder to the Aerotech motion system using a custom printed bracket, we installed a 1.8mm-nozzle heatcore, connected it to the water-cooling circuit and began testing. We printed a few simple geometries to ensure the controller could drive it correctly before printing a #3DBenchy benchmark part.

Figure F-2 shows the best #3DBenchy we managed to print during this initial testing phase. We printed it at 200% scale using 2.85mm-diameter EcoTough PLA from Filaments.ca. We used a

1.2mm layer height, 60mm/s print speed, 20% rectilinear infill and a heatcore temperature of 190°C (identical temperature in both heated regions). Print time was approximately 45 minutes, excluding file loading and machine setup times.

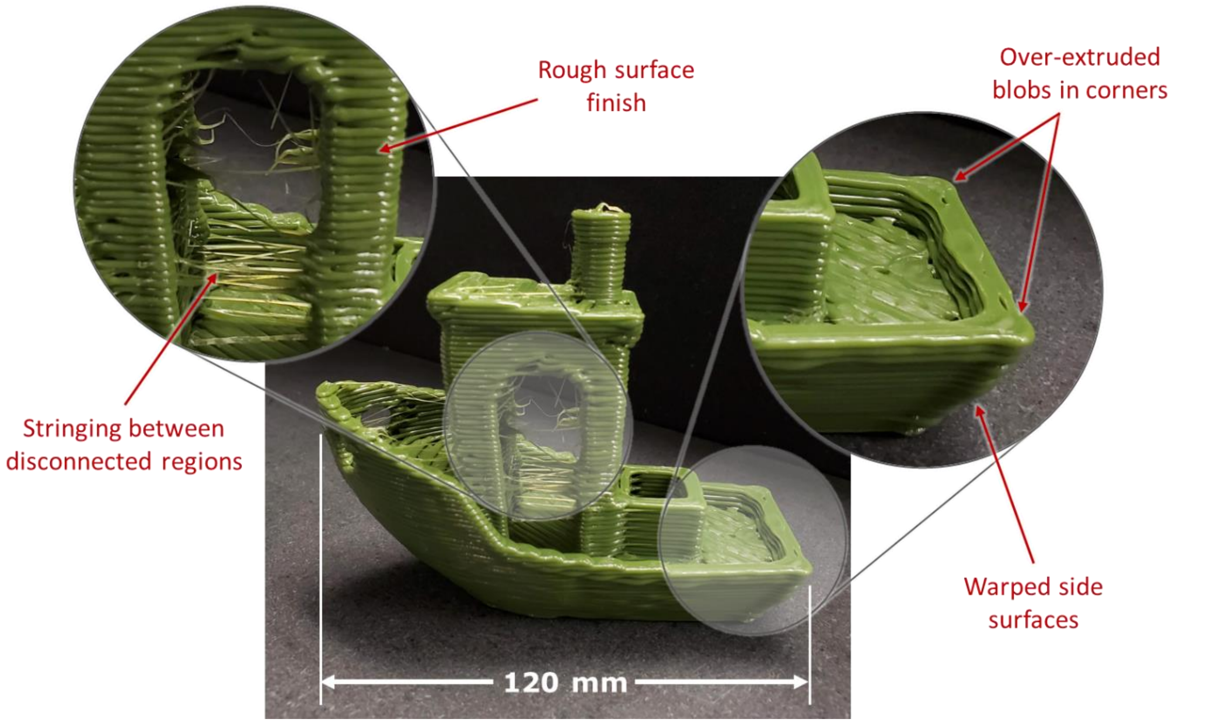

Rough surface finish
Over-extruded blobs in corners
Stringing between disconnected regions
Warped side surfaces
120 mm

Figure F-2: Main printing defects observed on the #3DBenchy printed with the Typhoon extruder on the Aerotech gantry system (1.2mm layer height, 60mm/s print speed)

Despite the significant improvement in part quality, the final part still contains many printing defects, namely:

1. *Stringing between non-connected portions of the part*. This defect is caused by the fact that we couldn't use filament retraction during travel moves due to limitations in how we sent extrusion commands to the printhead controller.
2. *Significant blobs in corners, despite using corner rounding.* Again, these defects are caused by limitations in extrusion management, which lead to motion acceleration and deceleration not being reflected in the amount of material being extruded.
3. *Slightly warped side surfaces*. These defects are mostly caused by insufficient part-cooling during printing, which leads to new layers being deposited onto material that hasn't solidified completely. Despite connecting the Typhoon's cooling shroud to one of the

laboratory’s compressed air outputs, setting it to 100psi (689kPa) – its maximum value – and slowing down the printing speed for short-duration layers, the narrow portions of the #3DBenchy still didn’t have enough time to solidify.

4. *Extremely rough surface finish*. Although this printing defect isn’t unexpected, it does show that the Typhoon is most likely better adapted to printing larger parts where surface finish doesn’t have as much of an impact on final part appearance. Unfortunately, the Aerotech gantry system’s limited z-height meant it wasn’t possible to print a larger #3DBenchy at this time.

Pulsar pellet extruder (beta version)

Installing the Pulsar extruder on the Aerotech gantry system was slightly more complex than for the other toolheads. Because of its size and weight, we chose not to mount it on a 3D printed bracket. Instead, we designed and machined a custom plate for it out of aluminum using a similar process to the one we used for the printhead controller’s enclosure (see Appendix D). Unlike the Typhoon and the production version of the Pulsar extruder, the beta version doesn’t offer part-cooling capabilities; we thus designed and printed shrouds for two radial fans to provide part cooling. Finally, we implemented a pellet-feeding loop to remotely supply the Pulsar extruder from a pellet reservoir outside of the enclosure. After printing a few square prisms to validate that all systems were working, we printed a slightly larger version of the #3DBenchy we had produced with the Typhoon (250% scale instead of 200%), as shown in Figure F-3a. After a few iterations, we printed our best part using a 1mm nozzle and Ingeo 3D850 PLA pellets from NatureWorks specifically formulated for additive manufacturing. We used a 0.9mm layer height, 40mm/s printing speed, and 185°C heater temperature (identical temperature in all three heaters). Print time was approximately one hour.

Figure F-3b shows the resulting #3DBenchy, which we can compare to the nearly identical part produced by the Typhoon extruder (see Figure F-2). Although we chose to print the two parts with different layer heights to better match their respective nozzle sizes, we notice the same types of defects on the Pulsar’s #3DBenchy as we had previously observed with the Typhoon. There is less warping on the part produced by the Pulsar extruder, which is likely caused by the smaller nozzle and layer height, as well as the lower printing speed. These printing parameters result in a lower volumetric flowrate which requires less cooling and time to solidify. The lower layer height on the slightly larger part also leads to a perceived improvement in surface roughness.

Perhaps unsurprisingly, stringing is worse on the Pulsar print, which features a much larger plastic melt zone. Because it extrudes pellets, the Pulsar extruder is also incapable of performing retractions, although this didn't make a difference here since retraction wasn't enabled for the Typhoon prints either. To limit stringing during travel moves, Dyze Design uses a spring-loaded ball inside the Pulsar's nozzle that is supposed to close its aperture when it stops extruding; however, this didn't work perfectly on the beta version of the extruder. Despite all these issues, we were enthused to see that printing with a pellet extruder – as well as several more conventional FFF printheads – worked reasonably well on the Aerotech system. We thus began the same initial testing process on the Fanuc robot.

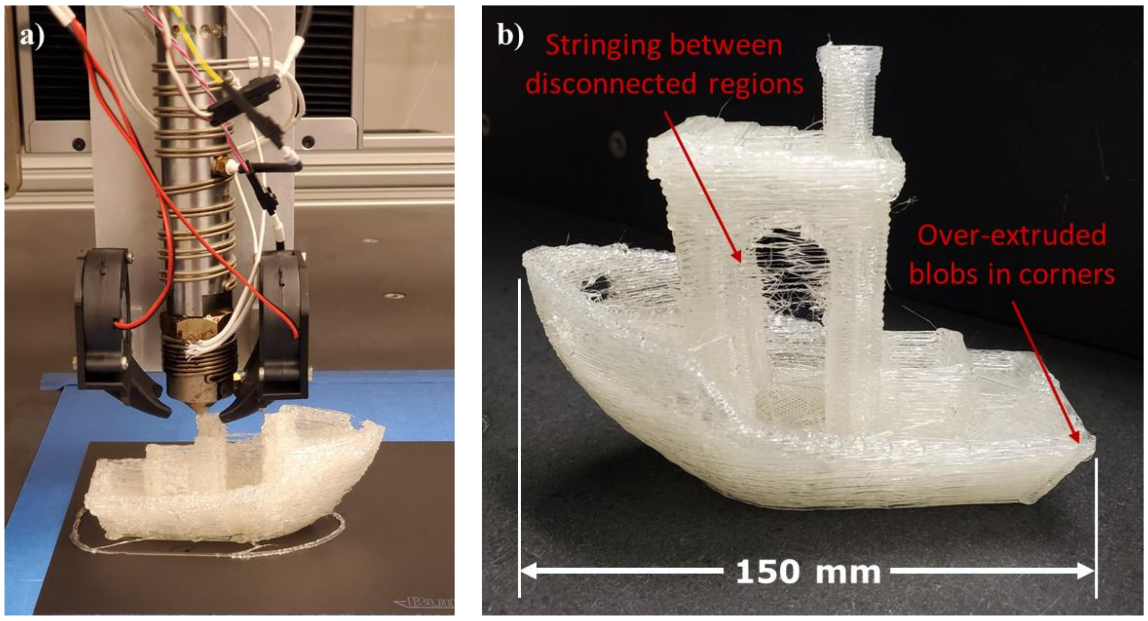

a)
b)
Stringing between disconnected regions
Over-extruded blobs in corners
150 mm

Figure F-3: a) Beta version of the Pulsar pellet extruder printing a #3DBenchy on the Aerotech gantry system b) Resulting part and main defects

APPENDIX G INITIAL PRINTING TESTS ON THE FANUC SIX-AXIS ROBOT

This appendix is analogous to Appendix F: it describes the initial troubleshooting we completed on the Fanuc six-axis robot using the static extrusion control method presented in Appendix E. We later developed the dynamic extrusion speed-control method described in Section 4.3 to improve print quality.

Typhoon extruder

As a first test, we printed the outline of a scaled-up tensile testing dog bone, which contains both straight and curved segments as well as multiple direction changes, features that had caused us issues during our initial printing tests on the Aerotech system. We used 2.85mm PLA filament from Materio3D, a 1.8mm nozzle, 1mm layer height, 50mm/s printing speed and 220°C heatcore temperature. Figure G-1 shows the resulting print. As expected, we observed the same types of errors as we had seen on the Aerotech system: blobs in the corners and inconsistent extrusion in curves. Unfortunately, these errors were even larger than on the Aerotech: at the end of every printed segment, there were gaps in the printed outline; at the beginning, the blobs were also even more obvious than what we had seen before. Curved segments were particularly inconsistent, and – unlike on the Aerotech system – blending moves together using the Fanuc's versions of corner rounding and move-linking didn't help.

To further test the system's performance in curves, we decided to print the first few layers of a large, single-walled vase. We downloaded the design shown in Figure G-2a from Thingiverse and printed the first 100mm of the vase scaled up 300% using the same filament and nozzle as before, a 0.8mm layer height, and a higher 250°C temperature to allow us to print at a faster 70mm/s print speed. The resulting print is shown in Figure G-2b. We observed two things: first, the print speed never reached 70mm/s (we discuss this issue in Chapter 7). Second, as expected, the resulting part outline was extremely inconsistent, with blobs appearing every few millimeters. This was the final factor which pushed us to implement dynamic extrusion control on the printhead controller.

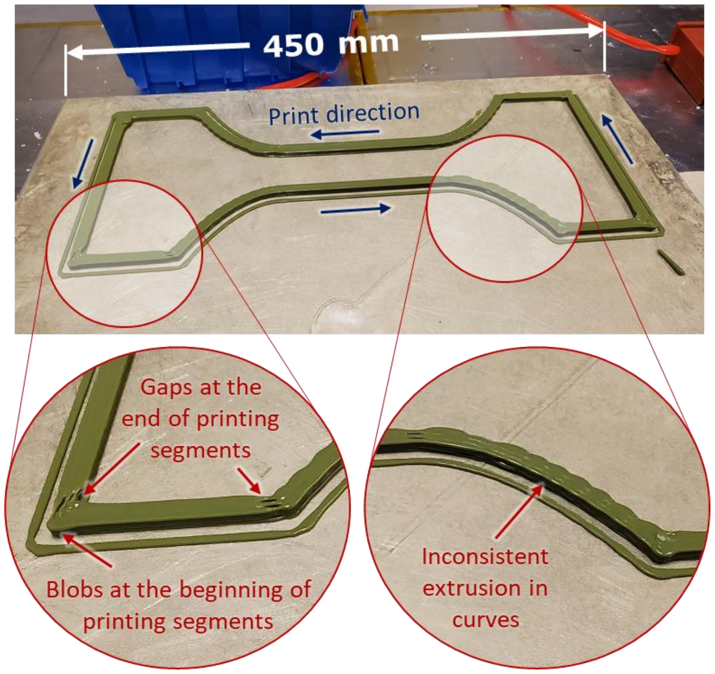

450 mm
Print direction
Gaps at the end of printing segments
Blobs at the beginning of printing segments
Inconsistent extrusion in curves

Figure G-1: Dog bone outline printed with the Typhoon extruder on the Fanuc robot

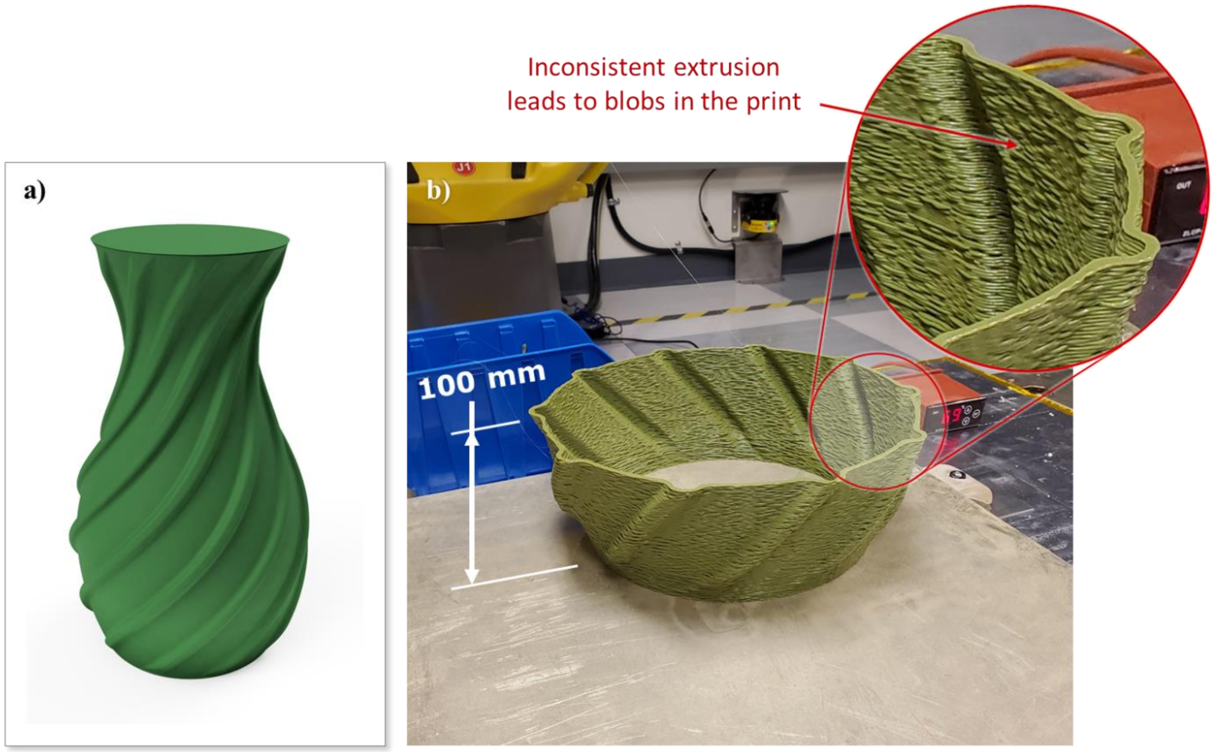


Figure G-2: a) Spiral vase by JJ76 [115] b) Printed with the Typhoon extruder on the Fanuc robot