Characterizing and Defending Against Cyber Security Vulnerabilities in Additive Manufacturing

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

in partial fulfillment

of the requirements for the degree

Master of Science

by

Bryan Gregory Kessel

December

2015

APPROVAL SHEET

The thesis
is submitted in partial fulfillment of the requirements
for the degree of
Master of Science

AUTHOR

The thesis has been read and approved by the examining committee:

Gavin T. Garner

Advisor

Carl Knospe

Barry Horowitz

John Thacker

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, Dean, School of Engineering and Applied Science

December
2015

Abstract

Additive manufacturing technologies represent the forefront of a modern industrial revolution. New machines such as 3D printers facilitate the design and effortless creation of part geometries that enable economic mass customization of manufactured parts. These new machines are therefore rapidly being adopted throughout the manufacturing world for the creation of both design prototypes and end-user parts. However, the increasingly widespread use and dependence on this emerging technology may pose new safety and security concerns to manufacturing, office, and home environments alike. Like other mechatronic systems, 3D printers employ software-controlled electrical signals to produce physical motions. Nearly all modern additive manufacturing machines incorporate an internet connection or at least have a direct connection to a personal computer with internet access, yet little attention has been directed toward cybersecurity solutions that could prevent malicious attackers from entering the system and manipulating the creation of parts. Unlike most other manufacturing processes (e.g. CNC machining), additive manufacturing allows a part to be constructed both internally and externally. It is therefore possible for a part's internal structure to be compromised in a way that is not easily detectable, even through close inspection of the external surface and other measurement techniques after fabrication.

The National Institute for Standards and Technology (NIST) has released an internal report detailing inherent security risks associated with replication devices such as 3D printers, stating that insufficient cyber protection exits for such machines. This research specifically addresses these concerns and suggests and verifies the efficacy of specific solutions. Several attack vectors have been identified through which a cyber attacker might be able to compromise the structural integrity of 3D-printed parts in ways that would not be easily detectable after the

part has been completed. These types of attacks were then emulated on a commercial-grade 3D printer, and their effects on the strength of the resulting parts were characterized using an Instron load testing machine.

Based upon ongoing DoD-sponsored U.Va research efforts regarding cybersecurity for cyber physical systems in general, the proposed solution for enhanced 3D printer security incorporates a highly-secured, trusted sentinel device monitoring the mechatronic system that operates the printer as a means for detecting potential cyber attacks. This research effort shows the feasibility of real-time detection of illogical printer behavior through the employment of a low-cost sentinel device that uses machine-independent sensors and transducers to monitor the machine’s motions and other reactions to instructions throughout the printing process. The results presented are intended to help stimulate the development of new security enhancements to protect 3D printers already at work in the field as well as current and future products in development.

Chapter 1 - Introduction

1.1 Motivation

The recent expiration of key patents related to additive manufacturing or “3D printing” technology has helped to fuel a proliferation of new low-cost 3D printers. Access to mechanisms and processes that were once proprietary has inspired companies such as MakerBot, Formlabs, Afinia, and 3D Systems to enter into the hobbyist 3D printing market with their own low-cost 3D printers. The wide distribution of such machines coupled with the relatively low cost of generic thermoplastic filament are challenging traditional plastics manufacturing techniques. Using additive manufacturing strategies, 3D printers can create individually customized parts as they are needed, reducing the wait time and infrastructure costs typically associated with small-scale production. However, the increasing use of 3D printing technology merits an investigation into its potential for malicious use. 3D printers are computer-controlled manufacturing devices that require no user control or supervision during the build process, and many of these machines are currently connected to existing networks and are supplied with internet access. If these machines are to be integrated into manufacturing facilities (either centralized or distributed), a system should exist to protect the integrity of the models coming from these machines and prevent or discourage malicious attacks on the machine or facility.

The 2010 Stuxnet attacks demonstrate the severity of cyber vulnerabilities in a mechatronic environment. Through these attacks the control systems for over 1,000 Iranian centrifuges used to enrich uranium were infected with a virus that caused the centrifuges to exceed their rotational design parameters and eventually fail over time [1]. Computer viruses such as these can cause major problems for industrial facilities wherein operators often have little understanding of the system’s architecture. An intruder can covertly gain access to the machines and cause them to create defective parts, damage the machines, or injure their operators and nearby workers. Additive manufacturing facilities are at an elevated risk for such attacks due to the potentially dangerous raw materials required for some 3D printing processes. These materials, such as highly-combustible powdered metals, pose significant threats to the safety of additive manufacturing facilities, machines, and operators. The recent explosion and subsequent fire at Powderpart, an advanced manufacturing facility based in Woburn, Massachusetts, illustrates this safety concern [2]. Vulnerabilities can be embedded into manufacturing equipment from local or remote sources, providing a mechanism to alter prints either randomly or selectively with a triggering device. This trigger could provide ease of entry into the system, providing full access to an unauthorized user at strategic times. Development of both preventative and detective techniques is therefore required for additive manufacturing systems.

The switch from traditional to additive manufacturing methods has enabled increased complexity of industrial cyber attacks. The key difference in additive manufacturing technologies is the ability to build hidden internal defects. With the application of 3D printing technologies to the fabrication of critical components, it is increasingly important to verify the absence of such defects in completed structures. With Chinese manufacturing facilities currently using metal 3D printing technologies to create central wing spars for Comac C919 passenger planes and front landing gear for the Chinese Navy’s Shenyang J-15 carrier-based fighter jet, cyber attacks could result in major loss of life [3]. Early failures in either of these situations could be catastrophic, and such examples would provide solid motives for malicious attackers during times of war. It is therefore important to understand the current vulnerabilities in additive manufacturing technologies and to address the existence of such weaknesses.

1.2 Background

1.2.1 Additive Versus Subtractive Manufacturing

The design complexity of working prototypes and end-use parts has long challenged the capabilities of traditional subtractive manufacturing techniques. Typically, increased feature complexity increases the difficulty and cost of production. The introduction of additive manufacturing (AM) technologies, however, has enabled the realization of complicated new shapes and mechanisms with geometries that were impossible to create using traditional manufacturing techniques. The ability to discretize features and join materials together via heat or binding agents allows the creation of much more complicated parts than those possible through subtractive techniques.

Additive manufacturing involves fundamentally different processes than traditional subtractive manufacturing techniques. A subtractive process such as a milling operation involves the removal of material from an initial material stock using a cutting tool. Excess material is removed during the operation, and when sufficient material has been removed from the stock piece, the operation is complete. Subtractive manufacturing processes like turning or milling fundamentally introduce kinetic energy into the work piece. Throughout the course of the operation, this kinetic energy is dissipated as waste heat or deformed chips, causing the temperature of the work piece to rise [4]. Additive manufacturing processes, however, do not require the removal of stock material during the operation, as only enough material is expended as is required to create the final model. Instead of starting with a material stock and discarding any unwanted features, additive processes start with raw materials and build parts by joining those raw materials together. The resulting part is a unification of raw materials made possible through clever manipulation of their material properties.

3D printing is growing in popularity due to its low entry cost and high versatility. There are virtually no limitations on the geometries of buildable features for 3D printers due to their stratified build process and relaxation of the need for special fixturing. The only major limitations with this process remain in the final microstructure, the types of material used in the procedure, and the surface finish. While researchers have been pushing to print a more diverse range of materials, some materials require special processes that require large amounts of energy or complicated machines to print, which in turn causes printing costs to escalate. These different technologies support different materials and each process has its own relative advantages and disadvantages. The main disadvantage for 3D printing processes when compared to mass production processes is that 3D-printed models can take orders of magnitude more time.

1.2.2 3D Printing Techniques

3D printing is a category of additive manufacturing (AM) techniques that encompasses several different fabrication technologies. This category contains such processes as Stereolithography (SLA), Three-Dimensional Printing (3DP), Fused Deposition Modeling (FDM), Selective Laser Sintering (SLS), Laminated Object Manufacturing (LOM), Color-Jet Printing (CJP), Electron Beam Melting (EBM), and Direct Metal Laser Sintering (DMLS) [5]. Machines that use each of these technologies are commercially available, but some of these strategies are limited to use with certain materials. SLA printing, for example, is limited to materials that solidify when exposed to UV light. EBM and DMLS printers are usually used with metal powders, while 3DP is generally used with any powdered material that can be bonded with an adhesive. FDM printers are some of the more robust printers, with the ability to print nearly any material that can be liquefied and solidified by heating and cooling elements. SLA printers are typically used to print photopolymers whereas FDM printers print thermoplastics. The low cost of polymer-based FDM and SLA machines has led to their wide-spread application for making plastic parts.

Stereolithography (SLA) falls under the broader category of Solid Free-Form Fabrication. This technology was invented in 1986 by Charles W. Hull, who later founded 3D Systems, Inc. [6]. Hull's patent on the process specifies the hardening of liquid resin via a hardening agent, either by an induced jet or by a bath of ultraviolet radiation. This application of ultraviolet radiation, used to instigate photopolymerization in the resin tank, can be controlled with extreme precision due to recent advances in laser technology and optics. This means that a typical SLA machine has a higher resolution than a typical FDM machine (due to the precision to which polymerization reaction can be localized). A typical SLA printer can achieve layer thicknesses of 50-100 μm and minimum feature sizes of 250-380 μm, while a typical FDM printer can achieve layer thickness around 180 μm with minimum feature sizes of 250 μm [7]. With extremely precise control of femtosecond laser pulses, some Micro-SLA machines have boasted feature sizes in the nanometer scale, such as the printed model of a bull shown below:

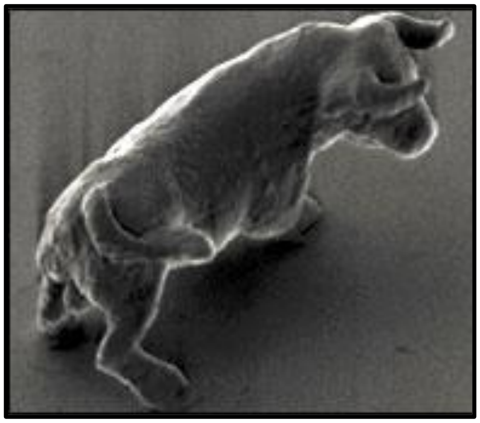


Figure 1 - A Model of a Bull Printed via Micro-SLA (10 µm in length)[8]

Even though SLA models have ultra-precise features, parts created from the FDM process tend to be more durable and more suitable for end-use applications [9]. These inherent differences come from the bonding processes used in SLA and FDM printers. SLA printers use a series of mirrors to direct a laser beam into a vat of photopolymer resin. A computer sends control signals to the laser, rastering the resin, resulting in photopolymerization reactions and thereby hardening the resin from a liquid into a solid [7]. SLA printers build their parts up in slices. To print each layer, a movable platform is positioned in the resin tank such that only a small layer of the resin exists between the platform and the previous layer of the part. The laser beam then cures each subsequent layer of the resin, causing it to bond to the previous layer of the part. The use of a laser as the hardening mechanism allows for very precise part formation. Since the part is submersed in liquid resin, it is possible for the excess resin to undergo photopolymerization, causing a loss of dimensional tolerance or increased surface roughness. This submersion makes the creation of support structures much more difficult as well. The bath of liquid resin restricts the use of a separate support structure. However, printers like the Carbon3D have found a unique solution for this support issue. Instead of using a separate photopolymer material for support structures, the Carbon3D forms voids in the resin vat through the injection of air bubbles into the vat. These controlled bubbles act as a support structure for floating features.

Like SLA printers, FDM printers also stratify the model, printing each layer individually and then bonding each subsequent layer to the last. Instead of using photopolymers, however, FDM printers employ thermoplastics, a class of polymers that are glassy at low temperature, yet can achieve viscous liquid-like flow properties at high temperature. With the addition of sufficient heat, a thermoplastic can be pushed through an extrusion nozzle by a drive mechanism. A typical FDM printer uses a gear drive system that pushes a strand of thermoplastic filament with a circular cross section (typically 1.75, 1.78, or 3 mm in diameter) into a heating chamber, as shown in Figure 2 below.

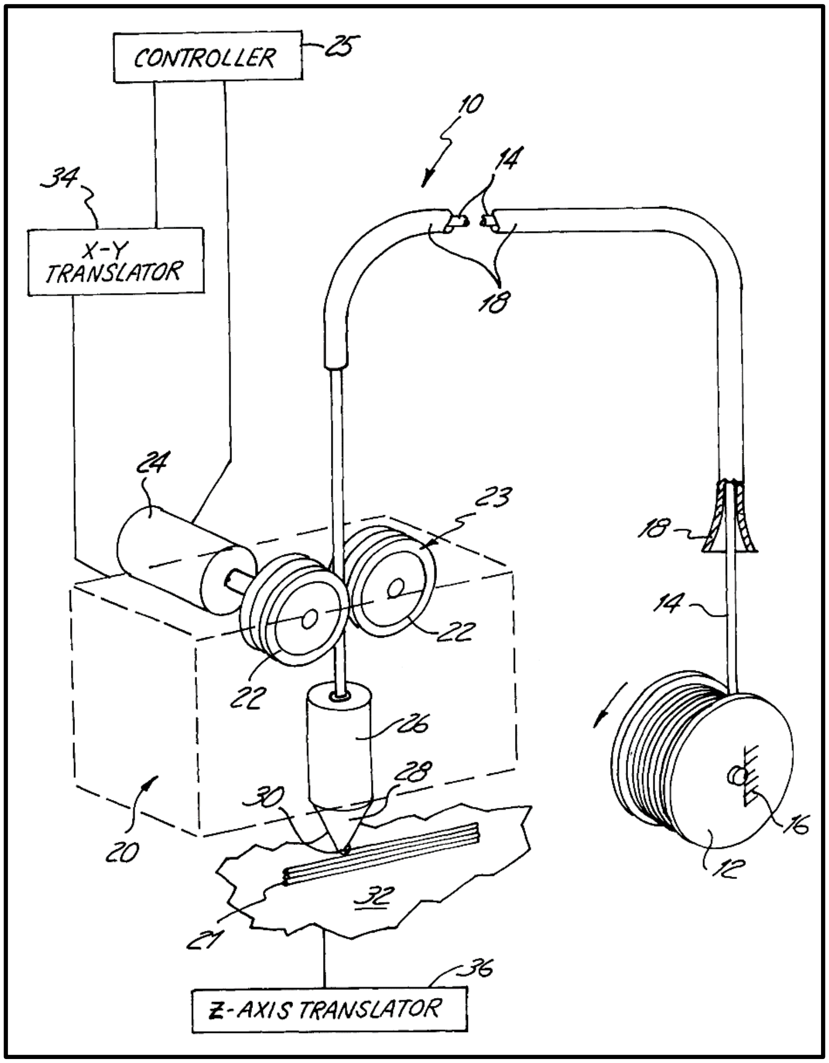


Figure 2 - Diagram of an FDM Extruder [10]

As a result of the Continuity Equation and the incompressibility of molten thermoplastics, the pressure created by the incoming filament causes liquid plastic to flow outward from the heating chamber, usually through a nozzle of a specified geometry. Nozzles are generally of circular cross-section, but vary in diameter for varying Z-resolution and bond strength. FDM printers can use a much more diverse selection of polymers than SLA printers, including several stronger, more durable polymers such as Acrylonitrile Butadiene Styrene (ABS) and Polylactic Acid (PLA). This research focuses mainly on FDM technology and its application with ABS thermoplastic because of the wide commercial application of FDM to the manufacturing of plastic parts.

1.2.3 Additional Information on Fused Deposition Modeling

The Fused Deposition Modeling (FDM) process was developed by S. Scott Crump, now the CEO of Stratasys, a major manufacturer of FDM and PolyJet printers [11]. This process utilizes melt-extrudable materials and is therefore used predominantly with thermoplastics, although it can utilize materials with alternative hardening mechanisms. Like the previously mentioned techniques, the FDM process slices the three-dimensional model into a series of sequential layers, converting each layer into two-dimensional toolpaths. A typical FDM printing mechanism uses a hot melt extruder to liquefy thermoplastic filament and deposit a molten ribbon of plastic in computer-designated locations on a build tray. Since the model is built in slices, each layer then thermally bonds to the previous layer of the workpiece. This thermal bonding is driven by the thermal energy stored in the incoming filament. When new filament comes in contact with the previously-extruded ribbon, molecular diffusion bonds the two layers at the interface, resulting in a solidified structure [12].

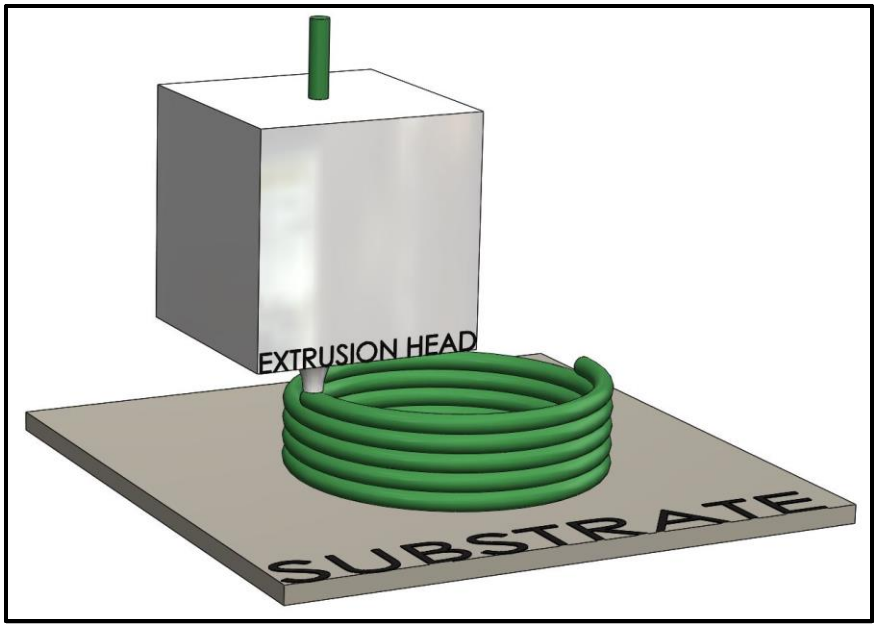


Figure 3 – Model of an FDM Extrusion Process

There are several problems that commonly arise during FDM build processes. The majority of these issues stem from the inherent heat cycling of the working material. When a layer of thermoplastic filament is deposited on the work piece, it must first be heated to a temperature at which it is easily extrudable. Since ABS is an amorphous thermoplastic, it does not have an established melting temperature. The plastic must therefore be heated high enough above its glass transition temperature to achieve desirable extrusion properties. When the extruder head adds this heated thermoplastic to the strata of existing material on the build substrate, the thermal energy contained in the newly-deposited filament drives interlayer mixing and subsequent bonding. The created structures are orthotropic composites of polymer filaments (partial bonding between filaments) and voids [13]. The discretization aspect of the FDM process drives the production of models with atypical material properties since complete mixing between printed layers is unattainable. The partial bonding between adjacent filaments causes anisotropic behavior within a given layer, while voids cause stress concentrations at their boundaries. Such stress concentrations are an integral weakness and can lead to premature failure of any part built with the FDM process.

1.3 System Aware Cyber Security

With inherent weaknesses in parts created by the FDM process, overall part strength becomes heavily dependent on the build parameters during the print job. Many commercially available 3D printers have minimal security measures to prevent or deter cyber attacker from manipulating these build parameters. Even though no major additive manufacturing facilities have yet reported being physically damaged by attacks to this date, it is important to consider the possible entry mechanisms into the 3D printing process that could allow cyber attackers to incapacitate additive manufacturing facilities or sabotage the parts they create. The Powderpart explosion is indicative of the potential for malicious attacks on additive manufacturing facilities and the scale of destruction that could ensue. In fact, the National Institute of Standards and Technology (NIST) has released a report detailing inherent security risks with replication devices like 3D printers, explicitly stating that insufficient cyber protection exists for such machines [14].

As part of this research effort, a solution is proposed involving System-Aware Cyber Security, which involves a “sentinel” device being integrated into the system for the purpose of monitoring machine operations while comparing those operations to a set of known valid operations for the process [15]. Such a scheme has already been applied to the operation of Unmanned Aerial Vehicles (UAVs) by Rick Jones and Barry Horowitz at the University of Virginia, where a microcontroller-based sentinel analyzed sensory input to determine the reliability of flight data sent from the UAV back to the base station during flight [15]. The application of a system-aware monitor (which will be called a “sentinel” in this research) to a 3D-printer could enable smart monitoring throughout the build process, allowing the printer/sentinel pair to determine if there is any deviance from a validated build process on the fly. Many 3D printers are internet-connected mechatronic systems with an on-board computer controller. None of these devices currently include system security measures to maintain unadulterated printer operation. One security option for the controller would be the addition of strict barrier security methods throughout the printer architecture (both hardware and software protection would be needed). This is an expensive, unreliable solution considering the vast array of possible cyber attacks. The concept of System-Aware Cyber Security proposes a much simpler solution: instead of securing all of the machine’s hardware and software, the security system needs only to secure the process itself. The security system should emulate the data transformation process performed on the printer’s circuit boards, predicting the machine’s physical outputs resulting from the data input to the printer. The sentinel therefore requires knowledge of the operation of the build process in order to properly defend against attacks. This monitoring device systematically detects breaches when illogical system outputs originate from logical data inputs:

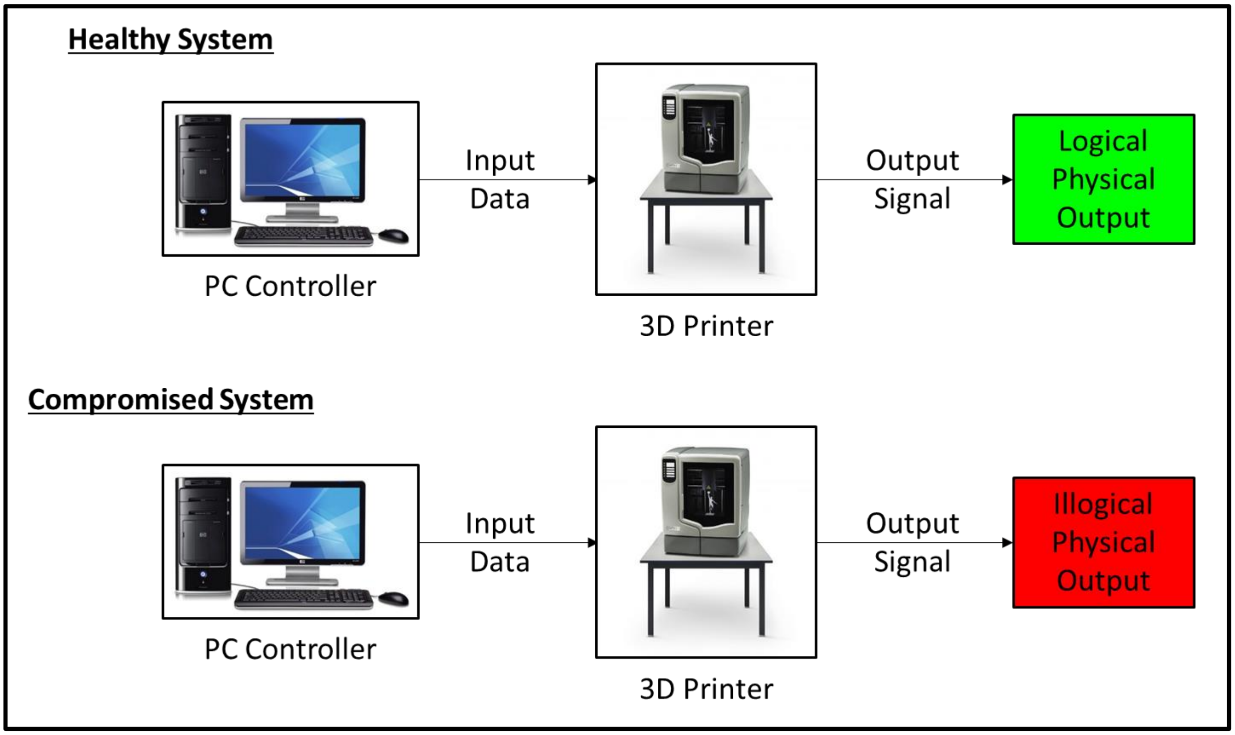


Figure 4 - Logical and Illogical Process Behavior

The advanced manufacturing context provides a perfect application for the newly developed sentinel monitoring technique. The sentinel can be interfaced with a 3D printer in such a way that cyber attacks to the printer do not allow access to the sentinel, allowing the sentinel to remain isolated from the main printer operations. For instance, the sentinel might read print parameter set points from its own memory chip. The printer might be allowed write access to this memory chip, but not to the sentinel's program memory. In such a setup, the sentinel is indeed purely a monitoring device; it does not impact and is not influenced by printer operation. Physical actuations can be monitored using low-cost sensors and transducers to give the sentinel real-time output monitoring capabilities. Additionally, the simplicity of the sentinel device's hardware makes it much more cost-effective to secure than the entire printer.

This thesis documents an additive manufacturing application of a System-Aware Cyber Security approach for embedded security measures. A prototype sentinel device was developed for the purpose of tracking the build process parameters for validation. Major attack vectors were also identified and emulated using a commercial-grade 3D printer to test the sentinel prototype's ability to detect any malicious manipulation of the printer's functions.

Chapter 2 - Identifying FDM Vulnerabilities

2.1 Selection of a Dimension uPrint 3D Printer for the Case Study

In order to further understand the implications of malicious attacks on 3D printers, cyber attacks were emulated through manipulation of both the build files and the 3D printer hardware. The machine selected for this study represented the majority of commercially-available FDM printers, employing commonly found hardware and software construction. This research project focused on FDM technology due to its widespread use and relatively low cost of entry, but the techniques used, vulnerabilities discovered, and solutions developed here are relevant for most types of 3D printers, including metal 3D printers.

The majority of the printers under consideration for this study were manufactured by Stratasys Inc., the leading manufacturer of FDM 3D printers. The models varied in price from roughly $2,500 to $40,000, with the Makerbot Replicator 2X at the low end and the Fortus 250mc at the high end of the price range. Metal printers were considered as well (both DMLS and EBM) but their high cost of entry prohibited their use for this particular project. The decision criteria allowed for the consideration of only industry-standard printers that might be found in a typical engineering prototyping lab or an advanced manufacturing facility. The less expensive Makerbot line of printers is most typically used in hobbyist applications, justifying its exclusion from this study. Furthermore, previous iterations of this printer have been open source (freely available to the general public) and used the common language of G-code to specify toolpaths. Attacking such a printer would be almost trivial, and it would not demonstrate the seriousness of this threat to commercial 3D printing systems. The Stratasys Dimension uPrint line, however, is frequently used in engineering prototyping environments, and was therefore a top contender.

The Fortus line of 3D printers was also considered for this study. A Fortus printer is a much more durable version of the uPrint and is most commonly used in advanced manufacturing facilities for large production runs of parts. Fortus printers receive the best mechanical and software components of all Stratasys FDM printers, but they function in almost exactly the same way as the Dimension machines, running the same type of proprietary toolpath files using similar motion interpolation processes.

It became apparent that entry-level models would not provide a representative sample of the 3D printer architecture commonly in use in manufacturing facilities and engineering design labs due to their relative lack of any complicated architecture. The choice was thus narrowed to the uPrint SE Plus and the Fortus 250mc. Further research discovered that the uPrint and Fortus printers differ mainly in their design software with very similar electromechanical hardware configurations (the Fortus does have feedback from encoders on the movement axes, while the uPrint only implements open-loop stepper motor control). The Fortus design software (Insight) allowed more complete control of the build parameters than the uPrint software (CatalystEX), including the ability to modify the in-fill pattern, road width (width of the extruded filament), and other location-related controls. The most notable parameter that was excluded from both of these software packages was control of the extrusion head temperature. Build temperatures were pre-set on the machine with a predefined optimal build temperature for each material.

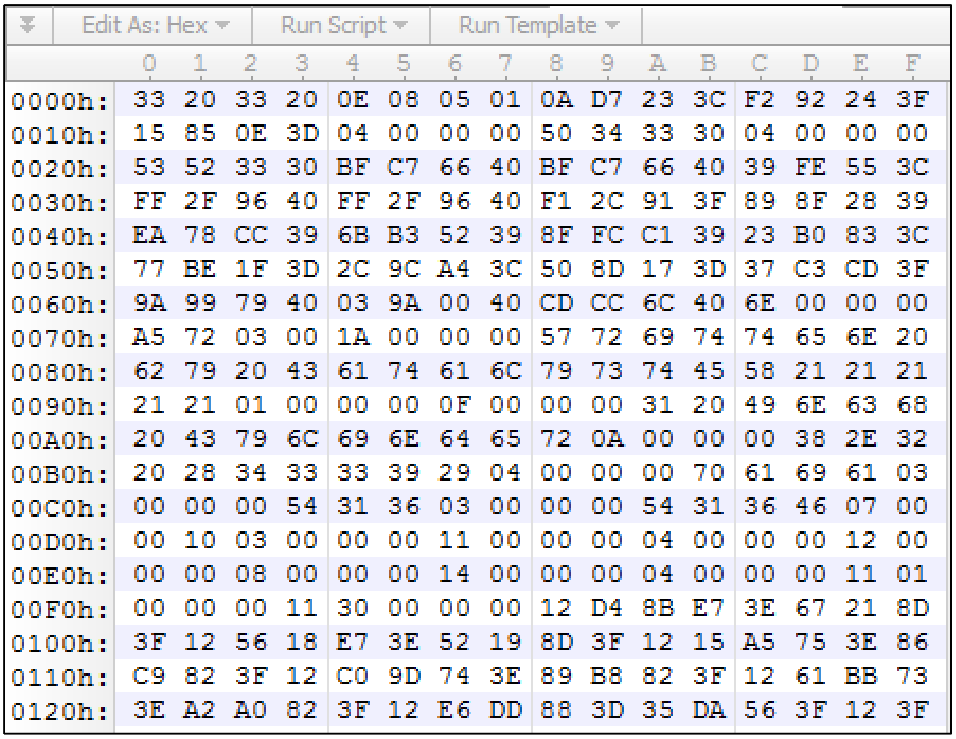
Both printers utilize the proprietary Stratasys “CMB” build file format. For the uPrint SE Plus, its CatalystEX control software converted the STL point cloud file into a series of machine-interpretable move commands that were then stored in the CMB file. The Fortus software (Insight) also creates a CMB file from a given STL file, but it allows more control over the generation and editing of toolpaths. Digging deeply into the CMB file format and deciphering it resulted in the discovery that all CMB files contained the same general information, regardless of the printer/software combination for which they were written. This enabled the manipulation of CMB files for use in different printers and allowed direct editing of certain key build parameters in order to emulate a cyber attack.

This discovery led to the decision to use a uPrint SE Plus for this research project since the same exact editing techniques could be extended directly to the Fortus line of machines, which cost several times as much. With the printer hardware selected, it was then necessary to dig further into the CMB file format to identify any weaknesses that could possibly be exploited by cyber attackers.

2.2 CMB Architecture and Vulnerabilities

All 3D printer build files are simply extensions of the traditional G-code files used for Computer Numerical Control (CNC) machines. A G-code file includes the relevant build parameters for the machine, such as spindle speeds, feed rates, etc. Along with these declarations, the G-code also contains a series of G-commands defining the machine's motion. For instance, a G01 command defines a linear movement, while a G02 command defines a circular arc movement. 3D printer build files are very similar if not identical to G-code, containing relevant build information followed by a series of move commands. These commands define the machine's step-by-step motion profile, ultimately creating a three-dimensional structure.

Attacking the build files of a given printer requires in-depth knowledge of the build file's architecture. Since the Stratasys CMB build file format is used for the majority of Stratasys FDM printers, it serves as a good case study for 3D printing build files. CMB files are unencrypted binary byte codes, with most data presented in four-byte longs. The architecture was interpreted with the help of the CMB Viewer program provided with Dimension uPrint machines. This program displays relevant print parameters contained in the CMB file as well as graphical representation of all toolpaths. CMB files were opened in a binary editor program and viewed in their hexadecimal representation to more easily understand their formatting. The following image is the raw data obtained from a CMB build file:



```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0000h: 33 20 33 20 0E 08 05 01 0A D7 23 3C F2 92 24 3F
0010h: 15 85 0E 3D 04 00 00 00 50 34 33 30 04 00 00 00
0020h: 53 52 33 30 BF C7 66 40 BF C7 66 40 39 FE 55 3C
0030h: FF 2F 96 40 FF 2F 96 40 F1 2C 91 3F 89 8F 28 39
0040h: EA 78 CC 39 6B B3 52 39 8F FC C1 39 23 B0 83 3C
0050h: 77 BE 1F 3D 2C 9C A4 3C 50 8D 17 3D 37 C3 CD 3F
0060h: 9A 99 79 40 03 9A 00 40 CD CC 6C 40 6E 00 00 00
0070h: A5 72 03 00 1A 00 00 00 57 72 69 74 74 65 6E 20
0080h: 62 79 20 43 61 74 61 6C 79 73 74 45 58 21 21 21
0090h: 21 21 01 00 00 00 0F 00 00 00 31 20 49 6E 63 68
00A0h: 20 43 79 6C 69 6E 64 65 72 0A 00 00 00 38 2E 32
00B0h: 20 28 34 33 33 39 29 04 00 00 00 70 61 69 61 03
00C0h: 00 00 00 54 31 36 03 00 00 00 54 31 36 46 07 00
00D0h: 00 10 03 00 00 00 11 00 00 00 04 00 00 00 12 00
00E0h: 00 00 08 00 00 00 14 00 00 00 04 00 00 00 11 01
00F0h: 00 00 00 11 30 00 00 00 12 D4 8B E7 3E 67 21 8D
0100h: 3F 12 56 18 E7 3E 52 19 8D 3F 12 15 A5 75 3E 86
0110h: C9 82 3F 12 C0 9D 74 3E 89 B8 82 3F 12 61 BB 73
0120h: 3E A2 A0 82 3F 12 E6 DD 88 3D 35 DA 56 3F 12 3F
```

Figure 5 - Stratasys CMB File Raw Hex Data

The CMB file consists of three main sections: header, toolpaths, and End-of-File (EOF) commands. The composition of these sections is shown below in Figure 6:

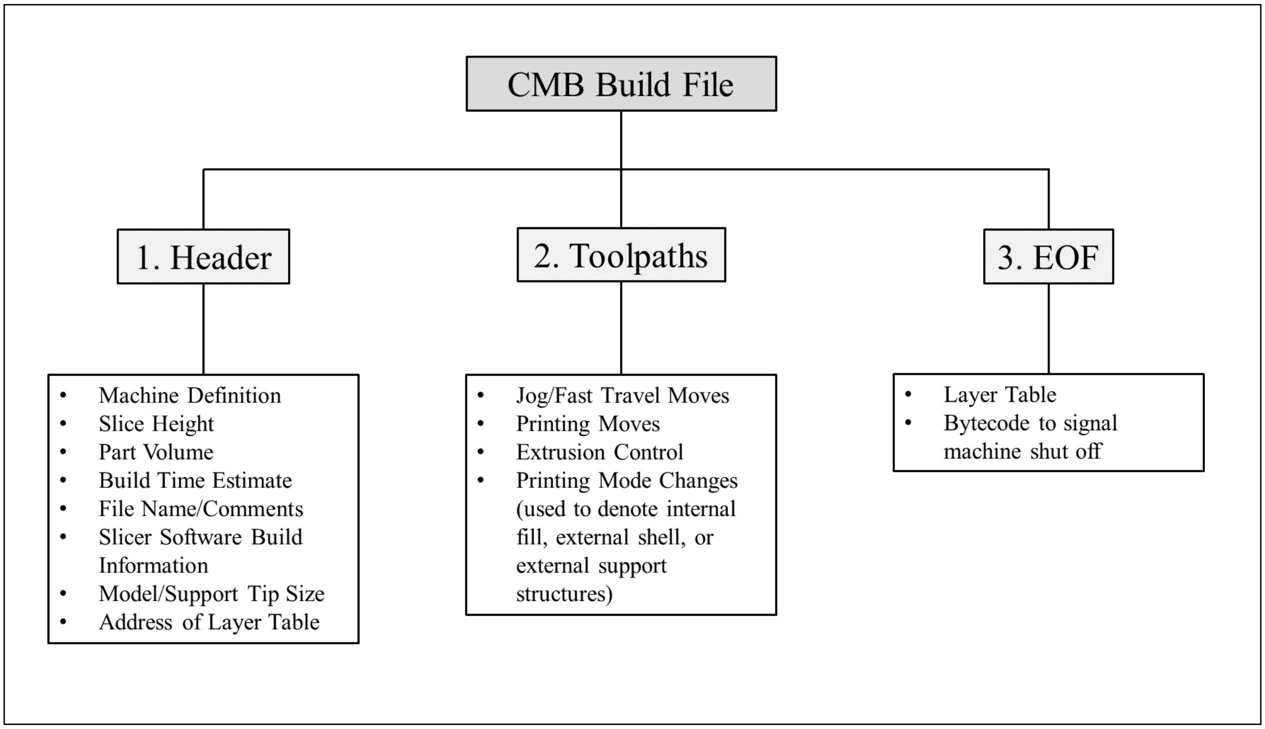


Figure 6 - Composition of a Stratasys CMB Build File

The header section contains the data displayed in the "Pack Details" section of the CatalystEX software during CMB file transfer and it is the information that the machine operator sees while preparing files for printing. The toolpath section details the movements of all four machine movements: x-travel, y-travel, z-travel, and plastic extrusion. Before each move is executed, however, the printer defines the "Print Mode", which contains definitions for which extruder to use (model or support), what part of the model is to be printed (interior fills, exterior shells, part roof, part floor, support raft, etc.), and other similar characteristics for the subsequent movement commands. Once the printer completes the designated toolpaths, the next instruction is the end-of-file. One component of the end-of-file section is the layer table. The layer table contains the addresses within the CMB file for each layer, stored as a four-byte integer. Each entry in the layer table points to the exact address of the beginning line of each layer, starting with the declarations of the bounding box for that layer (maximum and minimum layer dimensions). Once a layer is completed, the printer searchers the layer table for the address of the next layer. This continues until the machine steps through all of the layers in the layer table. If the number of entries in the layer table does not equal the number of layers declared in the header of the CMB file, the machine returns an error with the file. The end-of-file also contains a unique bytecode that the machine firmware designates as the signal to turn off the extrusion mechanism and to move the head gantry and the build tray to their respective rest positions. This move is a “presentation” movement to signify the end of the build process, finishing the instructions obtained from the CMB file.

Floating point numbers in the CMB build file are encoded using the IEEE 754 Binary32 system standard. This requires four bytes and is presented in little endian format. Any integers are also represented in four-byte, little endian format. Strings are represented with ASCII encoding, so each letter is represented with one byte. A string consists of a four-byte integer denoting how many characters the string contains followed by that number of ASCII-encoded bytes.

One noteworthy mention is that these CMB build files do not have any form of encryption. The data is therefore easily accessible, and once the proprietary encoding format is understood, editing these files becomes quick and easy. Individual movements or movement modes can be targeted to negatively affect the outcome of the print. Header data can also be discretely manipulated to provide incorrect process data for the print, such as pack outlines and build estimates.

2.3 Determining an Attack Strategy

In order to understand the dangers that cyber attackers pose to AM processes, one must first consider the build parameters at risk for malicious hacking. In their parametric study of FDM-printed ABS, Sood, Ohdar, and Mahapatra identified the following key process parameters that correlate strongly to a given part's dimensional accuracy and tensile properties [16]:

→ Tip Temperature and In-Process Temperature Gradients

→ Road Width

→ Fill Density (Air Gap)

→ Build Orientation

→ Raster Angle

With the exception of the temperature, all of these parameters are defined and stored in the CMB file, allowing direct manipulation without any specialized software. These build parameters demonstrate the fundamental problem with CMB file security: none of the information contained in the build file is encrypted. The security for the entire process is only as secure as its weakest point of entry, so each point of entry must be considered. Furthermore, even if the files were encrypted, they may still be vulnerable, but software security is a different branch of research that lies beyond the scope of this work and falls deep within the realm of computer science.

The identification of logical points of entry for the 3D printing process necessitates an understanding of the individual file transfers and transformations throughout the entire process. The process for a typical 3D printed part is shown below in Figure 7.

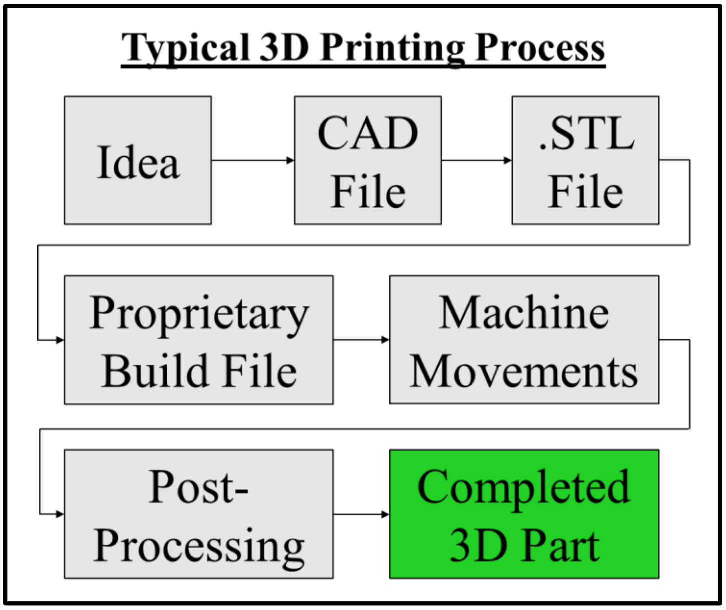


Figure 7 - 3D Printing File Transfer Process

The transformations represented above illustrate the points of entry where a hacker might interfere with the 3D printing process. A cyber attacker could deliberately build hidden defects into a part at several of these points. STL or build files pose the highest risk, as they are usually transferred between machines on external storage devices. These files could be intercepted and manipulated without the machine operator's knowledge. A group from the DREAMS Lab at Virginia Polytechnic Institute and State University has researched the effects of such software attacks on 3D printers by testing an STL interception algorithm on uninformed students and analyzing the results. They determined that changes to the STL file could go undetected by both the students operating the printers and the STL quality-checking software that verified the integrity of the model [17].

A more devious and covert attack methodology might target the interpretation of the build file once it reaches the machine. In the case of a Dimension uPrint, the CMB file is passed to an on-board move compiler. The move compiler reads the road settings from the CMB, calculates a suitable extrusion rate, and then saves the combined tool path and extrusion data into a Stratasys PCB file [18]. This move compiler is stored on the machine hardware in the Single Board Computer's Linux hard disk. An attacker with a basic knowledge of Unix-based operating systems could find the software for this move compiler and cause it to offset the machine coordinates for certain movement modes, or to change the extrusion rate calculation and therefore the heat flux from the heater block to the extruded plastic.

It is commonplace in most 3D printing techniques for models to need post-processing to remove support structures and extra material. An attacker could compromise a printed part by adding some agent to the post-processing chemicals that might corrupt the integrity of that part. For instance, many FDM parts are placed in a solution of sodium hydroxide and water (NaOH and H_2O) after printing in order to dissolve soluble support structures. The key to this bath is that the ABS model material is not adversely affected by the NaOH solution; however, addition of a third agent to induce semi-solubility of ABS is entirely possible. Monitoring the post-processing steps is therefore equally as important as monitoring the build process.

From a software viewpoint, the most vulnerable print parameter format is the CMB build file. STL files store a small amount of information regarding the location of model material, but STL files do not include printer metadata or the majority of the key print parameters. Additionally, some control software (like the Stratasys Catalyst EX and Insight programs) display the toolpaths created by the slicing algorithm, making the locational changes achieved by STL manipulation easily detectable by a competent operator.

Altering the firmware of the machine could represent another form of software attack, especially if the machine receives firmware updates over the internet, as do both the Dimension uPrint and Fortus machines. Modification to the Linux operating system that controls the uPrint is also possible. Access to the Single-Board Computer (SBC) located in the interior of the uPrint can be achieved either through an Ethernet connection or even by directly plugging a keyboard and monitor into the SBC. Physical access could be limited if the machine is locked or placed under video surveillance, but the firmware access could potentially be granted via the Ethernet connection to the Linux SBC.

The base assumption for this project is that an attacker could find a way to gain access to the machine’s hardware. This scenario represents an attacker modifying the machine’s firmware definitions or installing foreign hardware into the machine. Access to the machine enables a cyber attacker to adjust the road width, fill density, print orientation, tip temperature, raster angle, and other important build parameters. Certain build parameters can also be controlled by attacking the hardware of the machine and installing intermediate circuitry to adjust feedback signal levels. The tip temperature is a good example of one of these parameters.

For this attack emulation, a validated (unhacked) build file was sent to the sentinel while a modified (hacked) build file was returned to the PC-SBC interface software (CatalystEX). This modified build file displayed the same print file information since the CMB header remained unchanged. This means that the machine’s operator would not notice a difference during the file transfer process for a modified file or an unmodified file.

2.4 Covertly Modifying Build Parameters in a CMB File

Modification of the CMB build file allowed changes to part geometry through manipulation of machine toolpaths. For this use case, CMB files created with Catalyst EX were altered for the Dimension uPrint SE Plus FDM printer. Custom build files were also created for a Fortus 200mc FDM printer with the Insight design software. Using Insight provided much more control over the print parameters of a given 3D model since the software allows in-depth modification of nearly every machine movement. The Insight files were prepared for a Fortus 200mc since that printer shares the same build envelope dimensions as a Dimension uPrint SE Plus. All models for these tests were printed from Stratasys ABSPlus/P430XL ABS thermoplastic filament. The modifications of these build files are extremely difficult to detect if only slight changes are made to the file. Small changes to the build file can retain the same file metadata (file size, print estimates) while negatively affecting the part strength. These characteristics make build file modifications an obvious way to alter the mechanical properties of the final printed part.

Toolpaths were changed by locating the movement commands found in the section of CMB file directly following the header. Each movement command has the hexadecimal prefix "[redacted]" and a hex byte containing its motion descriptor. These motion descriptors are defined as below:

Note that the specific byte codes used by Stratasys FDM machines have been censored in this version of the document so as not to provide a roadmap for would-be cyber attackers to these machines.

Table 1 – Print Movement Motion Descriptors

Hex Byte Code	Movement Type
[redacted]	Jog/Fast Travel
[redacted]	2D Linearly-Interpolated Print (XY Plane)
[redacted]	Z-Height Change
[redacted]	Print with Road Width Alteration (Adjusts Extrusion Rate/Height Offset)

Each print layer generally contained multiple printing moves. This is due not only to the large amount of points that comprise a typical layer, but also to the shell/interior method by which the printer built models. A typical 3D-printed part contains five structures. The first structures printed on each layer were the support structures, beginning with a support raft. The floor, the dense bottom layer of the model, was built upon the support raft or a support structure built on the raft. After the floor came a combination of exterior shells and interior fills. The density of the interior fill could be controlled by the user in the CatalystEX toolpath creation. The Insight design suite allowed for control of the prismatic structure of the internal fills, while CatalystEX did not. Once the internal fills and the external shells for a given XY location had been printed, the roof was printed. As the final layer for a 3D-printed model, the printing of the roof signified the end of the build process. These structures are illustrated in the following figure:

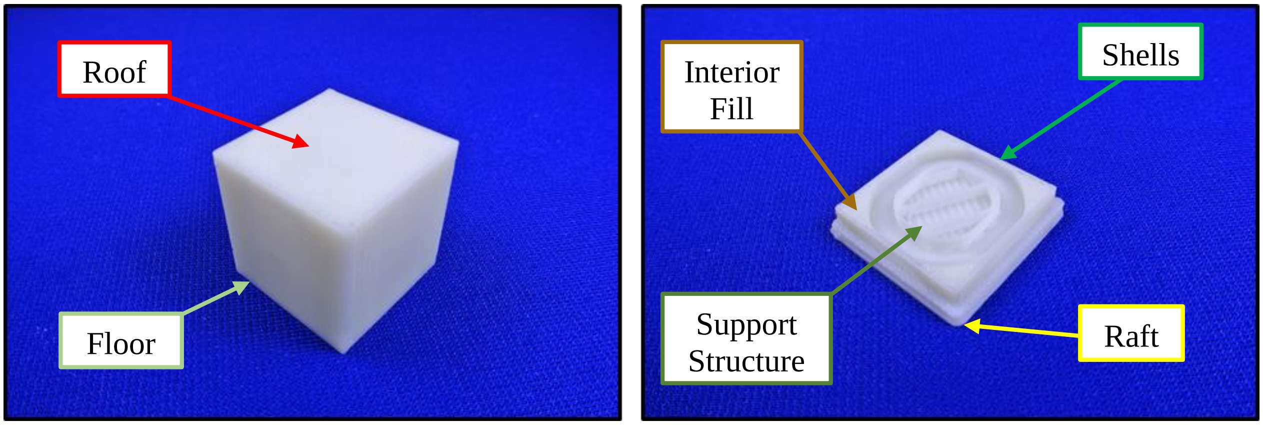


Figure 8 - Structures in a 3D Printed Model

The toolpaths in the CMB file had printing modes denoting which part of the model is being built by the current machine movements. This indicated to the machine which material to print (which heating element and extruder nozzle to engage). The following table displays the available printing modes for Stratasys FDM printers

Table 2 - Description of Printing Modes (Obtained from CMB Viewer Info Pane)

Hex Byte Code	Printing Mode
[redacted]	**"Part Interior"** – Used for shells on interior of model
[redacted]	**"Part Surface"** – Used for shells on exterior of model
[redacted]	**"Sparse Raster"** – Used for interior sparse fills of model
[redacted]	**"Solid Raster"** – Used for interior solid fills of model
[redacted]	**"Part First Layer"** – Internal fill of support material under raft
[redacted]	**"Part First Layer Perimeter"** – Shell of support material under raft
[redacted]	**"Support"** – Support structures not adjacent to model structures
[redacted]	**"Interface"** – Support structures adjacent to model structures
[redacted]	**"Sparse Bridge"** – Support structure placed between two interfaces
[redacted]	**"Support Bottom" –** First layer of the support raft

If cyber attackers attempted to alter a specific toolpath, first they would need to know the toolpath type. Since each printing mode requires a mode change, targeting specific printing modes is simple; one simply needs to identify the specific mode change and alter the ensuing toolpaths.

With this new understanding of the CMB file's architecture, it became evident that altering the geometric structure of the part would not be extraordinarily difficult. The most difficult attack would actually be the alteration of the tip temperature, since no software control of the tip temperature was available through the CMB file. This meant that software hacking of the temperature would need to happen through the machine's firmware. In order to emulate such an attack, the temperature feedback from the head temperature thermostatic control system could be intercepted and manipulated, thereby altering the extrusion tip temperatures in the same manner that a firmware attack could. Altering the temperature feedback for this closed-loop system caused the head to maintain a temperature unsuitable for proper extrusion, leading to either material jams or sub-par adhesion between material layers.

2.5 Attack Execution

2.5.1 Replace Fill Material with Support Material

The first attack targeted the internal fill pattern, replacing specific sections of model material with support material. A model was designed that adhered to the ASTM D638 testing standard for unreinforced and reinforced plastics. The model was built in the On-Edge (XZ) orientation as shown in Figure 9, but some internal fill toolpath modes in the CMB file were changed from the Solid Raster mode to the Support mode, changing the material type.

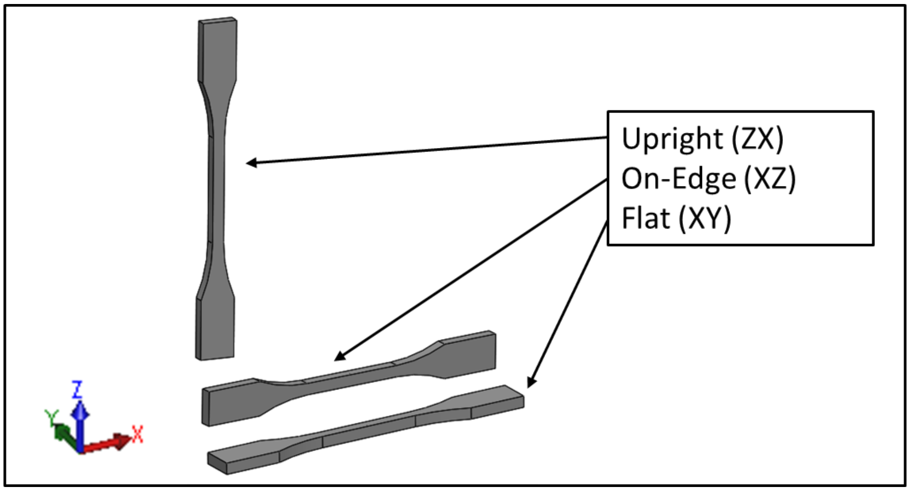
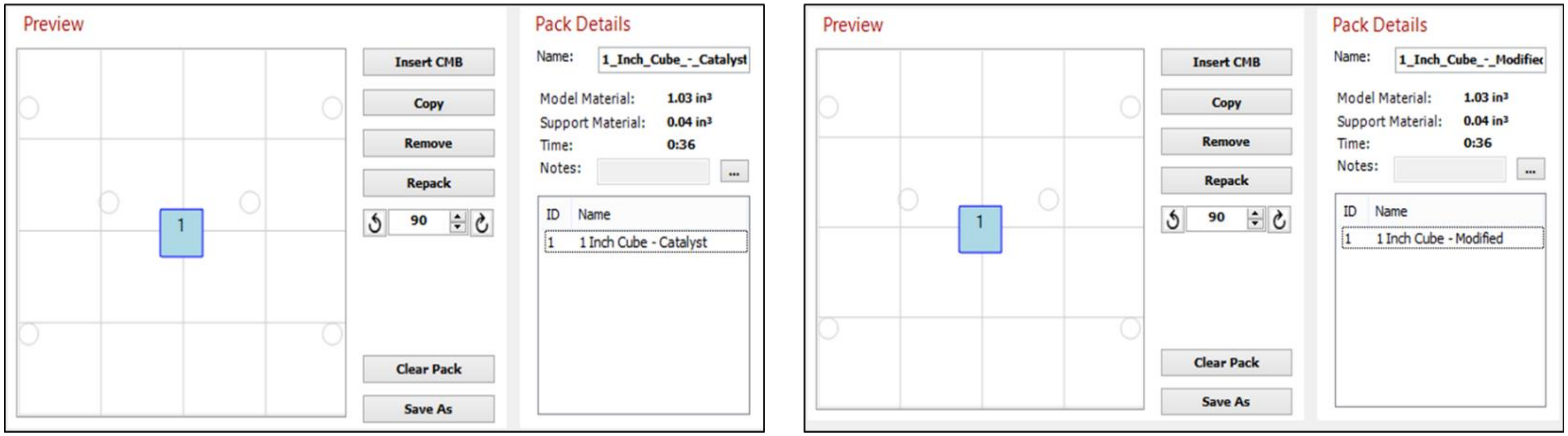


Figure 9 - Experimental Build Orientations [19]

The first step of attack emulation was locating the layer table within the build file. The entries in the layer table give the addresses for each layer, where a “05” byte is stored, identifying that the succeeding bytes define the minimum and maximum dimensions for that layer. Following these minimum and maximum dimensions is the definition of the print mode. A mode change to the Solid Raster mode signals an internal fill toolpath. By selecting this mode change and altering the mode definition, the waypoints are retained, though the print material usage changes. This attack can be achieved covertly with no change to material and time estimates through the clever manipulation of the print metadata as illustrated in the figure below. The figure shows two completely different models: the model on the left side is cubic, yet the model on the right side is cylindrical. However, the cylinder’s build file contains metadata from the square build file, thus incorrectly displaying the print information to the operator.

User View in Catalyst:



Name: 1_Inch_Cube_-_Catalyst

Model Material: 1.03 in³

Support Material: 0.04 in³

Time: 0:36

Notes:

ID	Name
1	1 Inch Cube - Catalyst

Name: 1_Inch_Cube_-_Modifiec

Model Material: 1.03 in³

Support Material: 0.04 in³

Time: 0:36

Notes:

ID	Name
1	1 Inch Cube - Modified

Figure 10 – Using the Same Metadata for Two Different Print Jobs to Hide Hacks (Left has toolpaths of a cube and right has toolpaths of a cylinder, yet both show cubes in the pack)

2.5.2 Fill Density Alteration

The internal density of these printed specimens also affected their mechanical behavior. When differing fill densities were used, the tensile test specimen attained a different equivalent stiffness. This lowered the amount of force required for a given elongation. The cross-sectional area of the model also decreased, so the new area had to be accounted for when calculating the instantaneous stress (mentioned in more detail later).

Fill density alterations were initially performed using a method by which the tool path section of the solid control model was replaced with those from a sparse-built model. This method produced undesirable results, however, as the material usage was drastically reduced. The method for this attack was therefore changed to produce a local change in density to reduce obvious changes in material usage. The local reduction in density was placed in the gauge section to place the stress concentration in the highest-stress area of the model, driving fracture there. This attack was achieved by altering the road width for internal fill paths in a certain area. The overall file size showed no major changes, as with the material usage estimates.

2.5.3 Introduction of Pre-Existing Internal Cracks

The performance quality of a given test specimen was drastically reduced when a small crack was introduced. Since an additive manufacturing process involves the placement of raw materials in desired locations, 3D printers enable hidden flaws to be placed inside of the part, such as small internal cracks. The strategic placement of a small crack can induce fracture at the crack surface and can also severely reduce the ultimate tensile strength and fatigue life of the final part. Such a crack could be deliberately placed in an area in which the part might exhibit high stress under loading.

Emulation of this attack was performed by identifying the internal fill of the test specimen in the CMB file, then removing a few internal nodes from the internal fill toolpaths while adding some internal perimeter toolpaths to create a clean crack line. This could be considered a covert attack because during processing, the operator would be shown the original build metadata with incorrect time and material usage estimates. The overall part volume was only altered by about 0.001%, which means that the impact on material usage is minimal. Detecting such an attack through primitive optical inspection or weight measurements would be virtually impossible. The pre-existing crack was placed in the center of the specimen to encourage crack propagation in the center of the gage section, which, of course, was where the sample ultimately broke (See Figure 23 and Figure 35).

2.5.4 Print Seam Re-Location

Another important print parameter contained in the CMB file was found to be the entry point of the plastic extrusion. By default, the uPrint begins and ends extrusion at the same specified location for each layer (for the ASTM test specimens). The phenomenon known as die swelling causes radial expansion of the print bead, depositing extra material at these entry and exit locations [20]. Since the uPrint begins and ends extrusion at consistent locations around the part’s perimeter, a print seam is easily visible on the side of most parts, including the ASTM test specimens. This seam relies on the polymer sintering reaction to coalesce the individual polymer particles, causing it to be inherently weaker than the continuous polymer filament found in the rest of the layer (see Figure 12 below) [21]. The misaligned polymer chains at this location also create a stress concentration that can cause fracture to occur. Thus, by moving the part seams to critical locations, an attacker can target a specific high-stress area and induce a brittle fracture (low strain to failure) there.

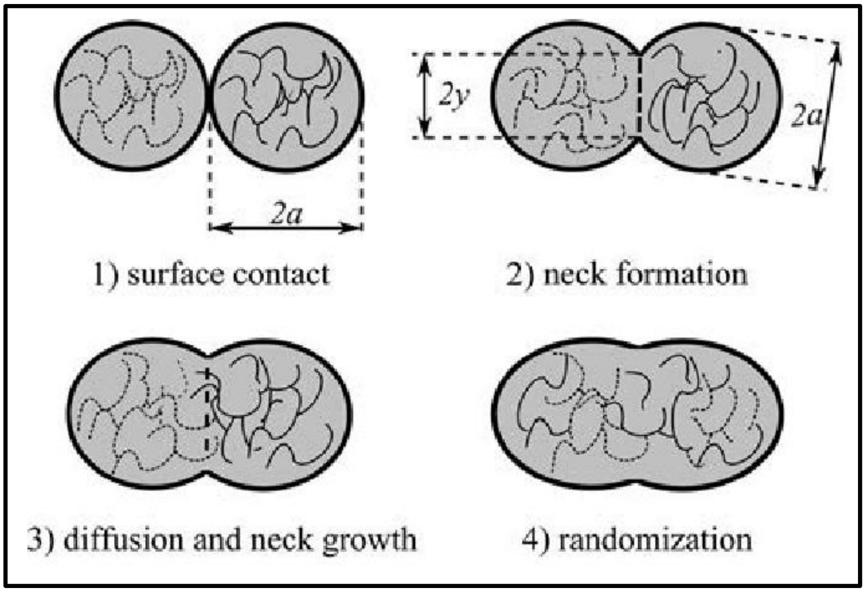


Figure 12 - Polymer Chain Mixing During the Sintering Reaction [20]

The location of these print seams was easily moved by changing each layer’s entry point in the Insight design software or directly in the CMB file. The CatalystEX software, however, did not have built-in support to adjust this parameter. This attack emulation therefore required the manipulation of a CMB file created in Insight such that it could then be imported into the CatalystEX for transfer to the printer. The standard ASTM D638 test specimen was built in SolidWorks, then saved as an STL file. The STL was then imported into Insight using the definition for a Fortus 200mc (same build envelope dimensions as the uPrint SE Plus) and all of the entry points were moved to the center of the gage section. Once the print seam had been aligned, the toolpaths were then exported to CMB format. This CMB file was then opened in a binary file editor and the header data and EOF lines were altered to agree with the format for uPrint SE Plus build files as previously displayed in Figure 6.

2.5.5 Tip Temperature Modification

The setpoint temperature of the extruder tip is the most important build parameter for FDM processes. The tip temperature controls the amount of thermal energy transferred to the filament as it passes through the extruder block, as governed by the following equation, given the assumption of constant heat flux [22]:

$$q = \dot{m}c_p(T - T_i) = \left(\frac{\rho v A c_p}{2\pi\left(\frac{D}{2}\right)L}\right)(T - T_i) \qquad \text{(Eqn. 1)}$$

where q is the heat flux, $\dot{m}$ is the mass flow rate of polymer through the liquefier, c_p is the heat capacity of the polymer, T and T_i are the entrance and exit temperatures of the polymer, v is the linear flow velocity, A is the cross-sectional area of the extruder, D is the diameter of the nozzle, and L is the length of filament in the liquefier [22]. The exit temperature of the polymer T_i is known as the extruder tip temperature.

Tip temperature is not a parameter stored in the CMB build file, so it cannot be manipulated using CMB alteration methodology. The Dimension uPrint SE Plus is designed to print with only two material cartridges: Stratasys ABS P430XL model material and SR430XL support material. This means that its temperature parameters are stored on the internals of the machine and the move compiler fetches them when it calculates the heater duty cycle and the extrusion rates. The persistent threat would therefore reside either as an embedded hardware device that alters the thermocouple feedback signals or from a firmware attack that alters the move compiler’s calculations for extrusion rates. For the purposes of this research, an interception of the thermocouple feedback was employed as the method of emulating a temperature setpoint modification attack. Stratasys Dimension FDM printers incorporate a temperature feedback loop to maintain strict control of the build temperatures. This closed-loop control system is shown in Figure 13 below. The attack involved the application of a feedback gain to the closed-loop control system.

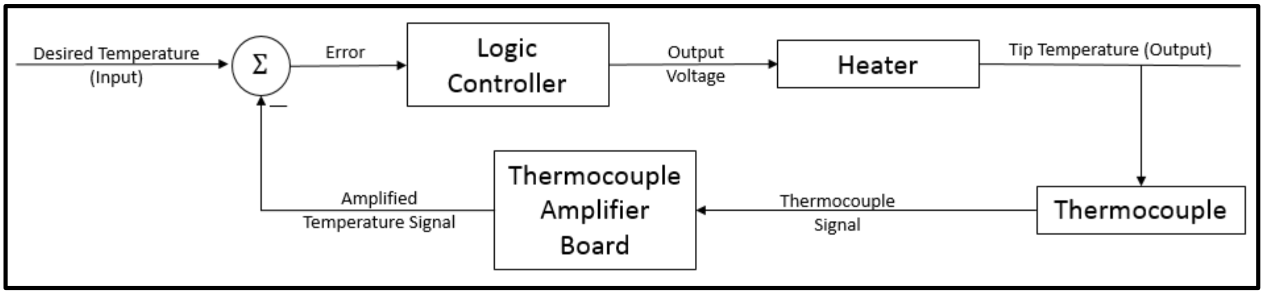


Figure 13 - Closed-Loop Temperature Control System

The gantry head of the printer contains several important circuits. The two main circuit boards located on the head are the head board and the thermocouple amplifier board. The head board performs key operations, such as regulating heater output signals and collecting extruder motor encoder signals. The thermocouple amplifier board serves three main functions. First, it contains a thermostat to send an alarm signal to the motherboard if the head overheats. It also contains a system of comparators wired as Schmitt triggers to determine if there is an open circuit in one of the thermocouples. Lastly, it contains Analog Devices AD597 thermocouple amplifier chips to amplify and linearize the feedback signal from the thermocouples in the head and build environment. These integrated circuit (IC) chips scale the temperature signal to an ice-referenced analog voltage range with a slope of $10\,\frac{\mathrm{mV}}{^{\circ}\mathrm{C}}$. There are three thermocouples in the uPrint: one in the model heater block, one in the support heater block, and one in the build environment. Each of these thermocouples is wired to a unique thermocouple amplifier chip, which then sends an output signal back to the printer's motherboard.

In order to manipulate the build temperature, the feedback loop needed to be cut and a device inserted to either amplify or divide the thermocouple's temperature signal. For a given K-type thermocouple, the thermocouple signals are on the order of 0 to 10 mV for a 0 to 300°C measurement, while the amplified temperature signal is anywhere from 0.2V to 3.5V. Since this signal is scaled by the AD597A IC chips, the temperature signal sent to the motherboard is between 3.0V and 3.1V when the head is heated sufficiently for extrusion. It was thus decided that instrumentation amplifiers or voltage dividers would be more suited to adjust the output signal of the thermocouple amplifier board than to its input signal.

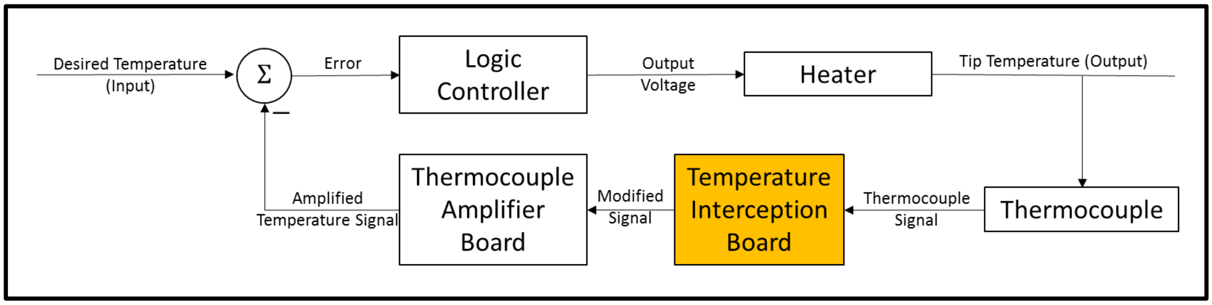


Figure 14 - Block Diagram for the Temperature Interception Board

The Temperature Interception Board (TIB) was therefore strategically placed to alter the output of the thermocouple amplifier board before it reached the motherboard, as shown in Figure 14 above. The TIB contains circuits for both multiplying and dividing the voltage, with a non-inverting amplifier to provide a gain of greater than 1 and a voltage divider circuit to provide gains less than 1. If the gain is set higher than 1, then the temperature feedback to the motherboard will be larger than the actual temperature in the head. Therefore, the motherboard will lower the head temperature until the temperature feedback comes down to the set-point temperature, which is 300°C for the support heater and 310°C for model heater. The circuit diagram for the Temperature Interception Board is shown below:

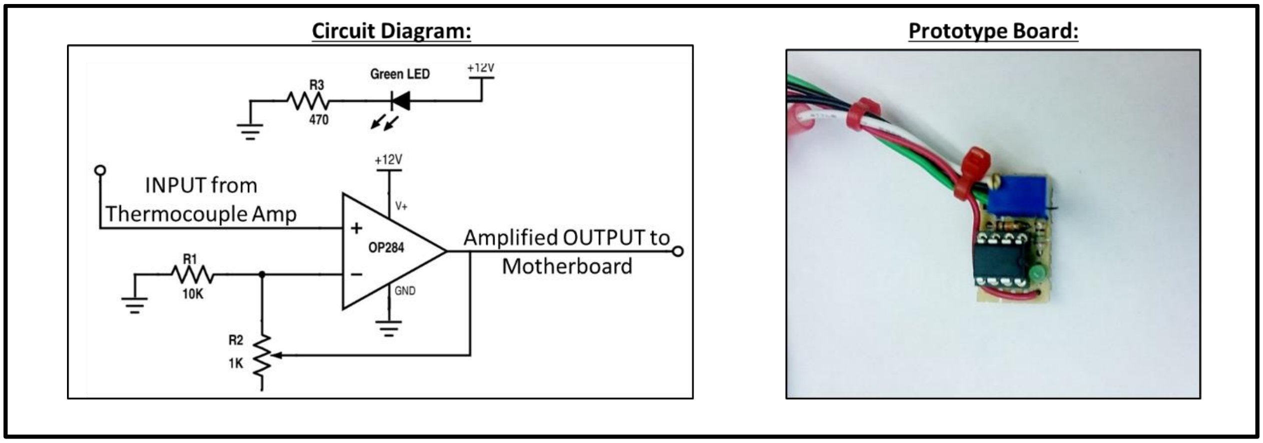


Figure 15 - Prototype TIB Amplification Circuitry

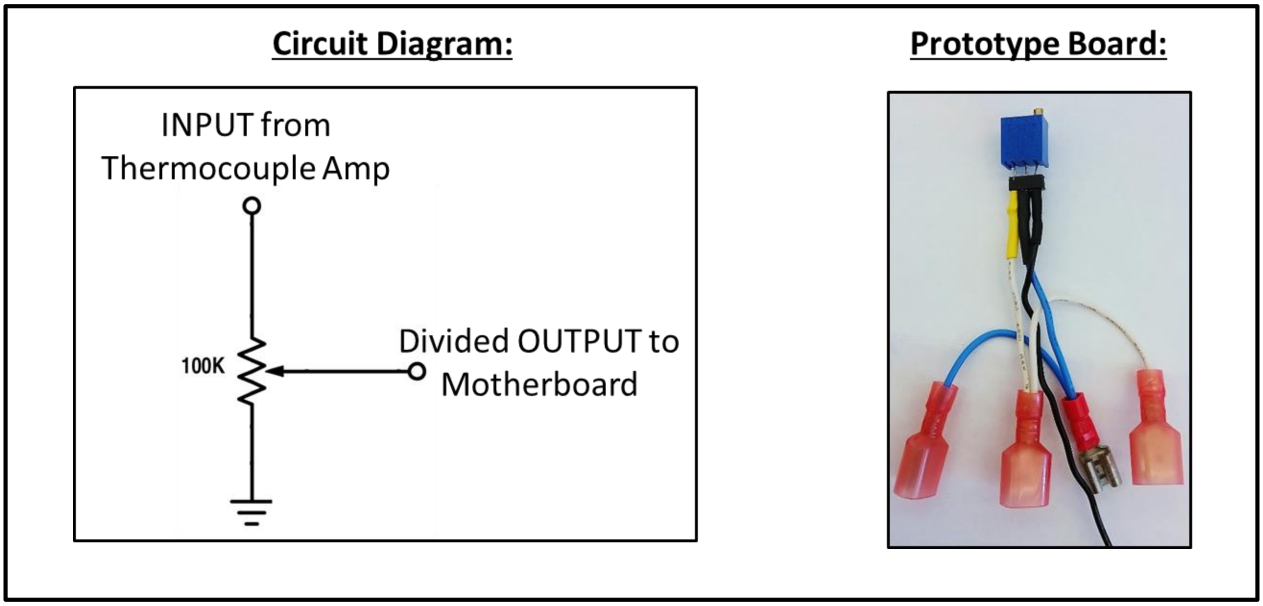


Figure 16 - Prototype TIB Division Circuitry

2.6 Detection

Comprehensive human or video monitoring of typical 3D printer build processes would be very difficult. With plastics, even small voids can compromise the mechanical properties of the printed model, resulting in lower stress and strain to failure. These voids can be caused by deviations so miniscule that even a machine operator staring at the printer during its operations might not detect any noteworthy process anomalies, as demonstrated by the temperature alteration attack. Some printers such as the Objet Connex can print in multiple locations at once, further complicating the monitoring process. The difficulties associated with validating printing parameters throughout the entire build process lead to the alternate solution of creating a robust sentinel monitor to simply observe the printer's movements and operations.

Chapter 3 - Characterizing the Effect of Attacks on Mechanical Properties

3.1 Preparing Samples for Tensile Testing

The effects of these attacks on the integrity of 3D printed models required characterization to provide a better understanding of the motivations behind potential attacks. The mechanical properties of specimens containing the results of these various types of attacks were assessed by measuring the difference in ultimate tensile strength and elongation at break between unaltered control samples and deliberately compromised specimens.

3.1.1 Choosing a Control Sample

Control specimens were chosen from the three cardinal print orientations described by Figure 9 to exhibit the characteristic strength reductions for each attack case. Due to the weaker flexural delamination strength of printed ABS parts, only the XZ and XY orientations were considered for testing the first four attacks. The candidates for the control specimen are shown below:

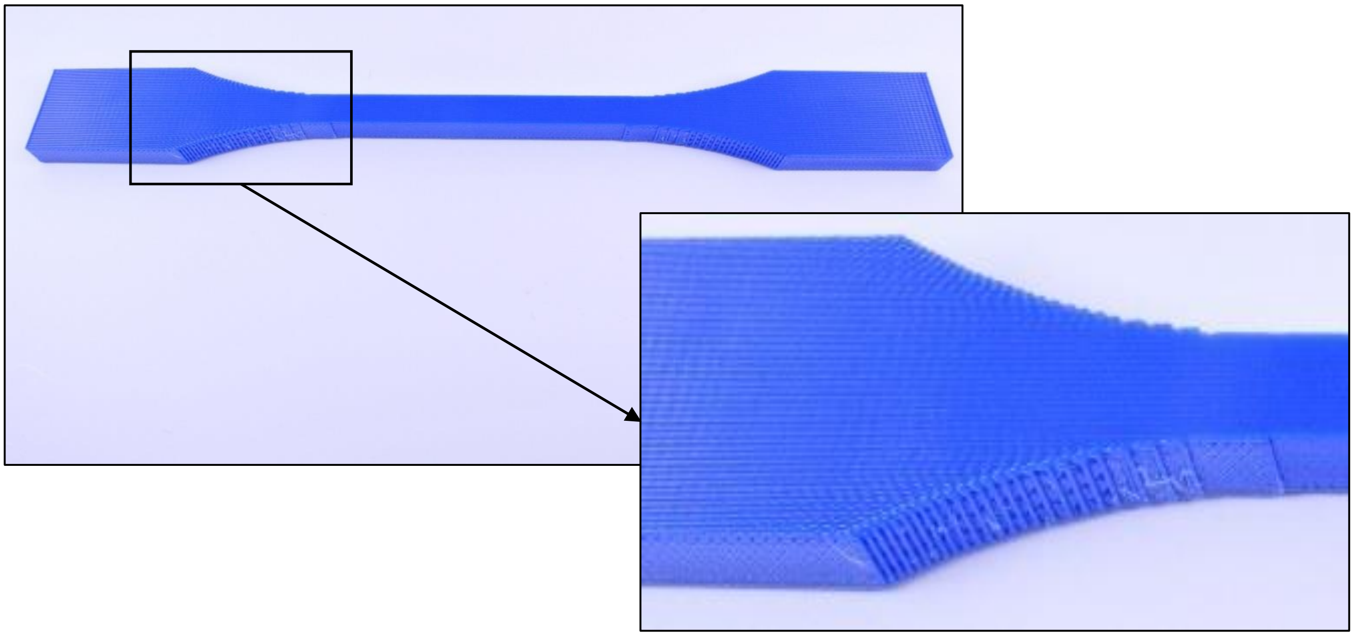


Figure 17 – ASTM Sample - On Edge Orientation (XZ)

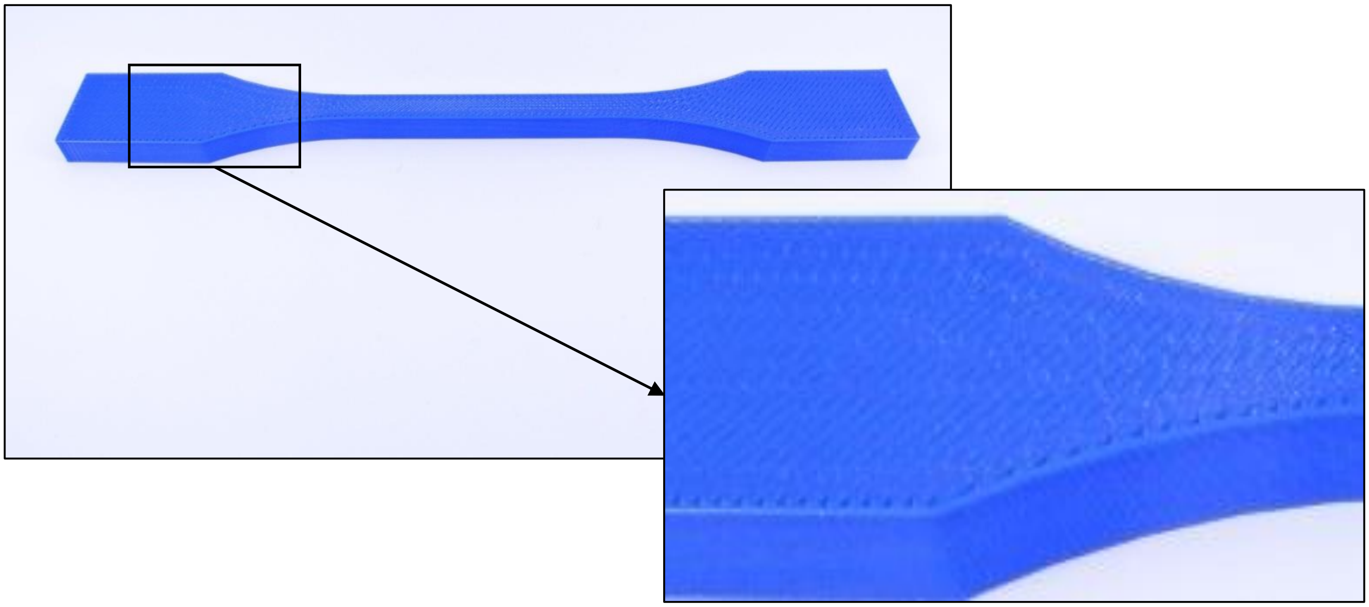


Figure 18 – ASTM Sample - Flat Orientation (XY)

The specimens printed in the XY orientation exhibited an undesirable fracture behavior, with all specimens breaking outside of the gauge length as shown in Figure 19. The XZ specimens, however, produced a favorable fracture behavior, with each specimen breaking at random locations.

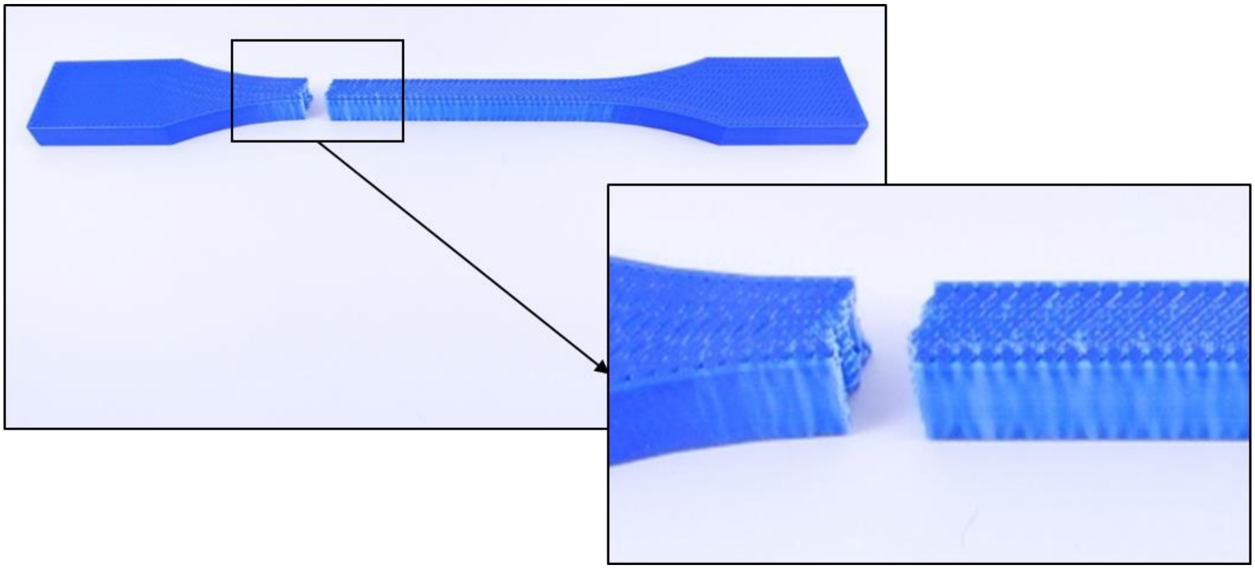


Figure 19 – Undesirable Failure Behavior of the Broken Flat (XY) Solid-Filled Sample

This unfavorable fracture of the XY orientation specimens was due to a concentration of air pockets in the radius of the print near the fracture surface, as shown in Figure 20. The curved section contained large air pockets along the wall of the part due to discretization of the internal raster pattern. This introduction of air gaps into the internal fill created a pre-existing crack whose existence caused failure to occur at that point. The reduction of area due to these air gaps resulted in a stress concentration, inducing a slightly higher stress at this location than elsewhere in the part. This Flat Orientation (XY) specimen was therefore rejected in favor of the On-Edge (XZ) specimen for the experimental control.

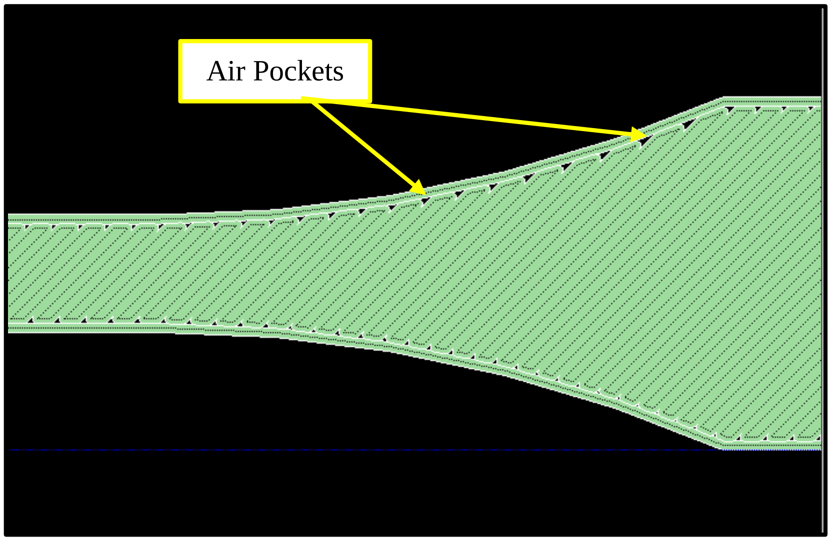


Figure 20 - Toolpath Simulation of an FDM-Printed Test Specimen (Obtained Using Stratasys Insight Design Software) – Notice the Air Pockets Created by the Raster Pattern along the Top Edge

With the control specimen chosen for the validation case, compromised specimens were printed containing flaws that represent each of the five attacks described in Chapter 2. These attacks are listed in Table 3 below:

Table 3 - List of Attack Vectors

Attack Vector	**Description**
1	Switch to Support Material During Interior Fill Paths
2	Alter Print Density Within Interior Fill Paths
3	Turn Off Extruder During Interior Fill Paths, Leaving Small Cracks
4	Add or Move Part Seams in High-Stress Areas
5	Adjust Extruder Temperature During Printing

3.2 Printing Compromised Samples

It was determined that five specimens would provide a large enough sample size to account for any normal printing variances. The specimens for each sample were printed in packs of six to allow one extra sample for potential misprints while still building all specimens correlating to a specific attack at the same time. The unbroken attack specimens are shown in the figures below:

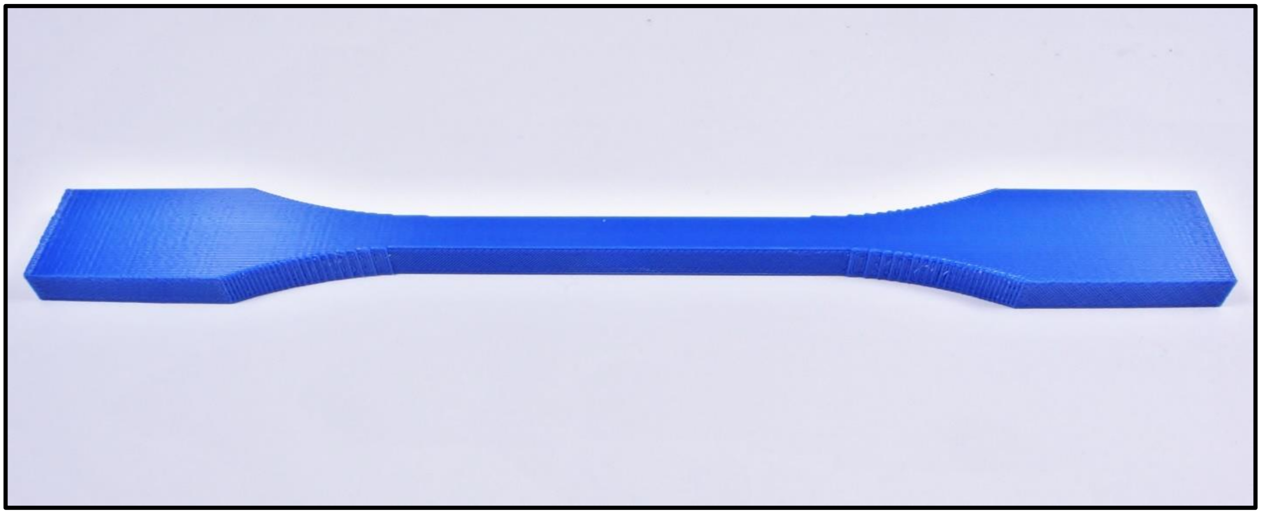


Figure 21 - Attack 1 - Replacement of Interior Fill Material with Support Material

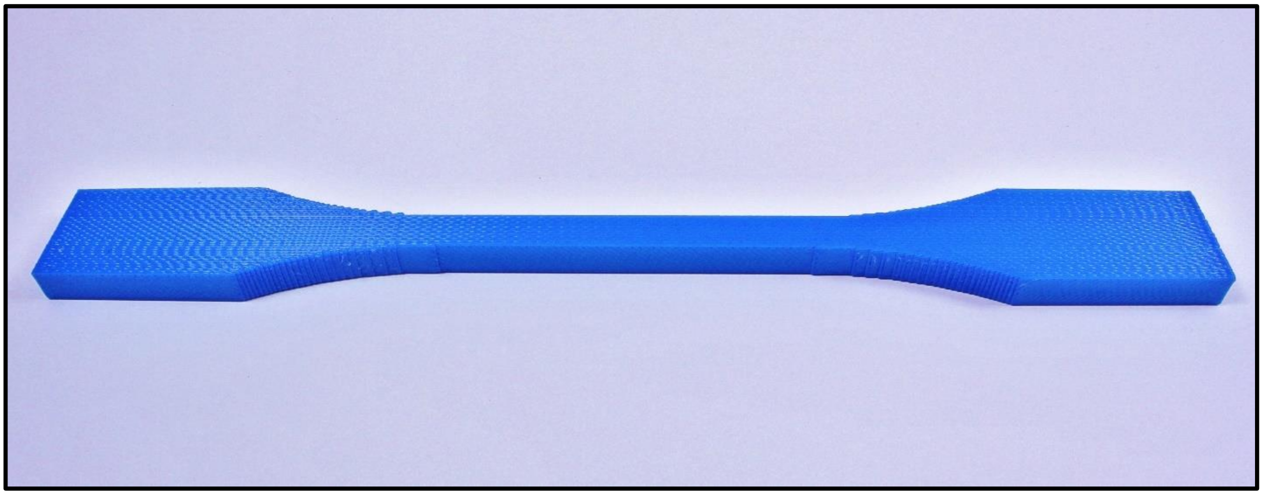


Figure 22 - Attack 2 - Localized Reduction in Internal Fill Density (More Detailed Picture of Internal Structure Coming)

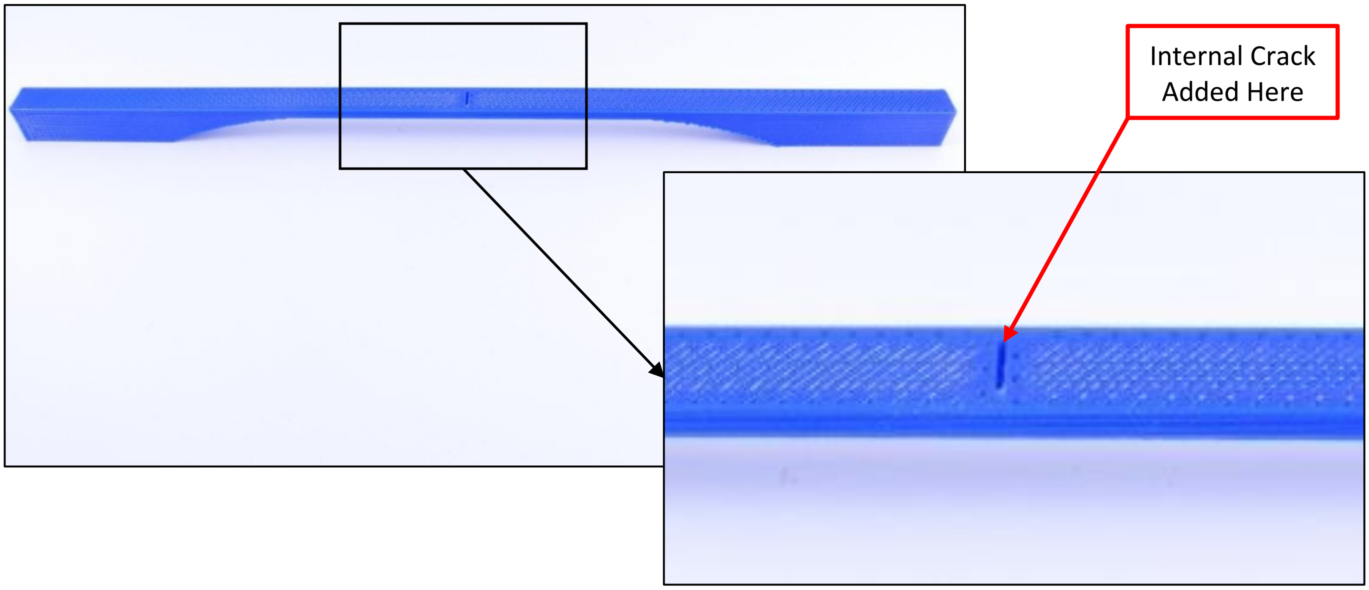


Figure 23 - Attack 3 – Introduction of Pre-Existing Internal Cracks

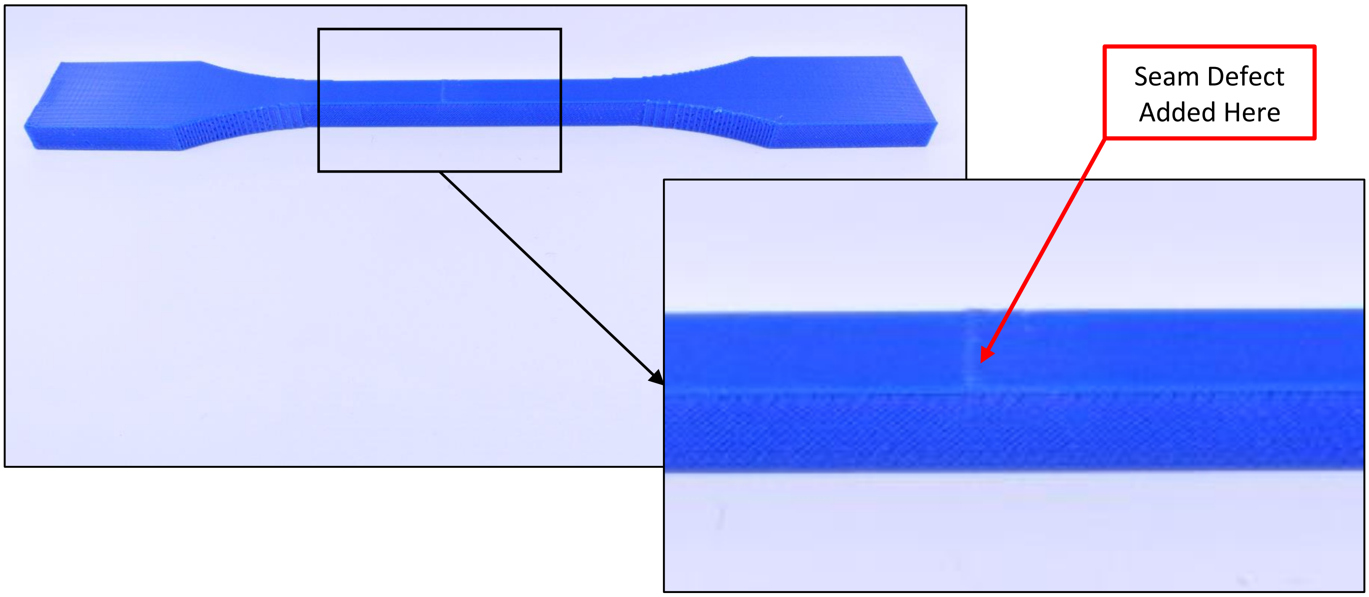


Figure 24 - Attack 4 - Print Seam Re-Location

Because of the thermally-induced sintering reaction between polymer layers, a different test orientation was employed for the fifth attack test to magnify the effects of altering the temperature of the extruder tip. The upright (ZX) orientation shown in Figure 9 was used for this test to illustrate the significant effect of the extruder temperature on the flexural delamination strength. When samples printed in this orientation were loaded in pure tension, the applied load separated the filament strands, straining the bond between adjacent layers.

The upright orientation was the best orientation to use for this test because of its tendency toward delamination under tensile loading, but the standard dimensions for the ASTM D638 Type II test specimen exceeded the build envelope of the Dimension uPrint SE Plus. The sample was therefore redesigned to fit inside the printer's build envelope (8" x 8" x 6"). The same general shape of the test specimen was retained, but both the grip length and the gauge length were reduced to shorten the grip-to-grip length. The figure below illustrates the difference between the two specimens:

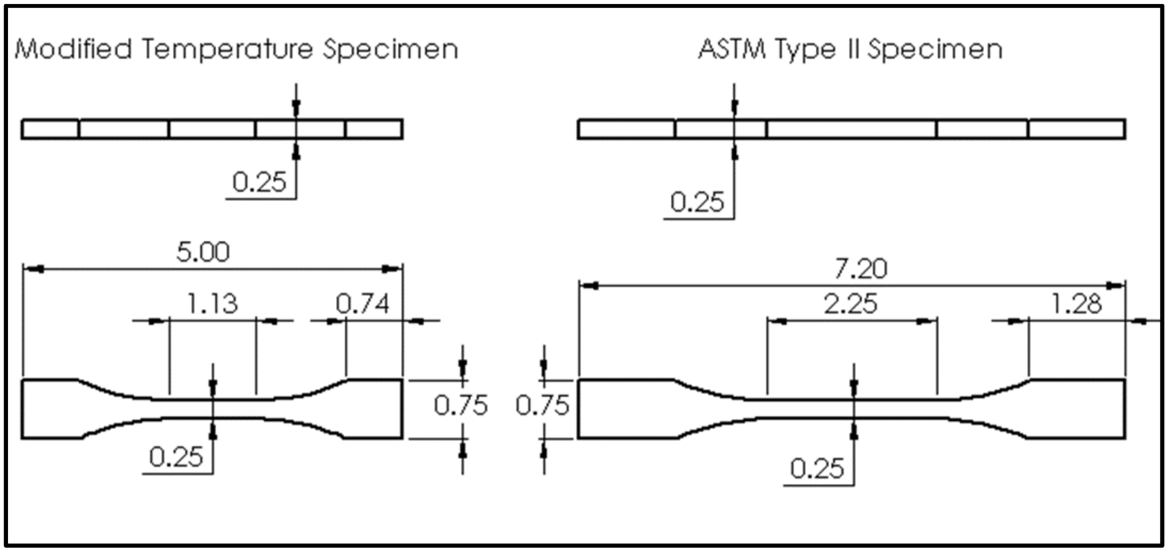


Figure 25 - Engineering Drawing of Modified Temperature Specimen (MTS) Compared to Control Specimen

All of the temperature specimens were printed according to the geometry of the Modified Temperature Specimen (MTS) shown above. They were built with a solid in-fill in the upright (ZX) orientation as per Figure 9. For this test, the tip temperature was altered from the default set point of 210°C. Temperatures were modified using a combination amplifier/divider circuit as described in Section 2.5.5. Temperature and elapsed time data were monitored and logged using an NI 9217 RTD measurement card installed in a National Instruments cRIO data acquisition device. The compromised build file produced the following specimens:

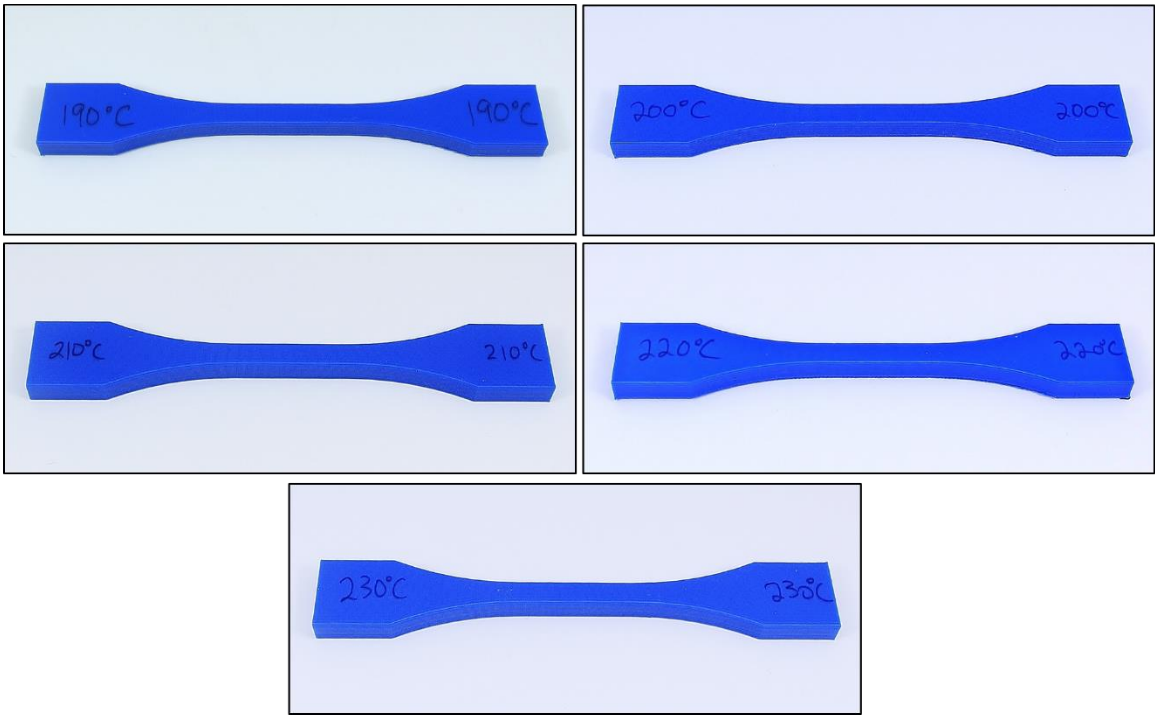


Figure 26 - Sample of Specimens Printed at Altered Tip Temperatures

The images above illustrate the difficulty for optical detection of temperature hacks to 3D printed models. All of the test specimens exhibited the same external characteristics, yet they had micro and nano-scale differences in the alignment of their polymer chains due to insufficient energy addition for polymer mixing. The temperature specimens were printed with the assistance of a precise temperature monitoring system, but some testing parameters were altered for the tensile test (crosshead velocity, failure criteria).

3.3 Tensile Testing Procedure

The samples described in Section 3.2 were strained to fracture in an Instron 5848 Micro Tester load testing machine fitted with a laser extensometer and an Instron ± 2 kN static load cell. Initial testing was performed using D638 test specimens, as this test procedure applies to thermoplastic materials [23]. During preliminary testing it was discovered that the standard Type I specimen in this setup regularly broke outside the gauge section. A Type II specimen was therefore employed, per the ASTM D638 testing procedures [23]. The Type II specimen has a narrowed gauge section, which exhibits a more significant difference in area between the grip section and the gauge section (larger radius of curvature). Use of the Type II specimen, along with moving the standard part seam from within the gage section to the end of the part, produced clean breaks in the gauge section for all parts. The Stratasys white paper discussing tensile testing of P430 ABS also excluded this print orientation from testing for undisclosed reasons [19].

Each specimen was elongated at a rate of five millimeters per second except for the temperature-adjusted specimens from Attack 5, which were elongated at a rate of one millimeter per second (due to the smaller overall specimen length). This extension rate was chosen to encourage fracture between 30 seconds and five minutes of the start of the test, as per the ASTM D638 standard [23]. Data was sampled from the load cell and extensometer at a rate of 200 Hz. This data was then recorded in CSV format (comma separate values) and processed through several custom MATLAB scripts (discussed in Section 3.3.1).

Laser tags were placed on the ends of the gauge section of each specimen prior to testing. These tags were aligned with the laser extensometer's beam to measure the extension in the specimen. This was achieved by taking an initial measurement for gauge length, zeroing the extensometer, and then measuring the displacement between the two laser tags. The measured displacement, δl, was used to find the engineering strain in the part, calculated as follows, where l_g is the gauge length prior to testing and ϵ is the engineering strain in the gauge section:

$$\epsilon = \frac{\delta \mathrm{l}}{l_g} \qquad \text{(Eqn. 2)}$$

All measurements of gauge length, extension, and strain were obtained in millimeters.

Engineering stress can be easily obtained as well, using the Instron ± 2 kN load cell. The load measurements were recorded in Newtons. The cross-sectional areas of the specimens were found by measuring the samples before fracture with calipers. To account for void inclusions during the printing process, effective cross-sectional areas were used for the stress calculations. These effective areas for the specimens were calculated using a mass fraction approach under the assumption that model density is linearly proportional to gauge cross-sectional area. As demonstrated in the figures below, the void inclusions during printing can significantly alter the cross-sectional area of the fracture surface. Effective areas were calculated for both the solid-filled specimens and the sparse-filled specimens for this reason. The notch and local density attacks, however, show the apparent stress in the model, so the effective area for the control specimen is used for those calculations.

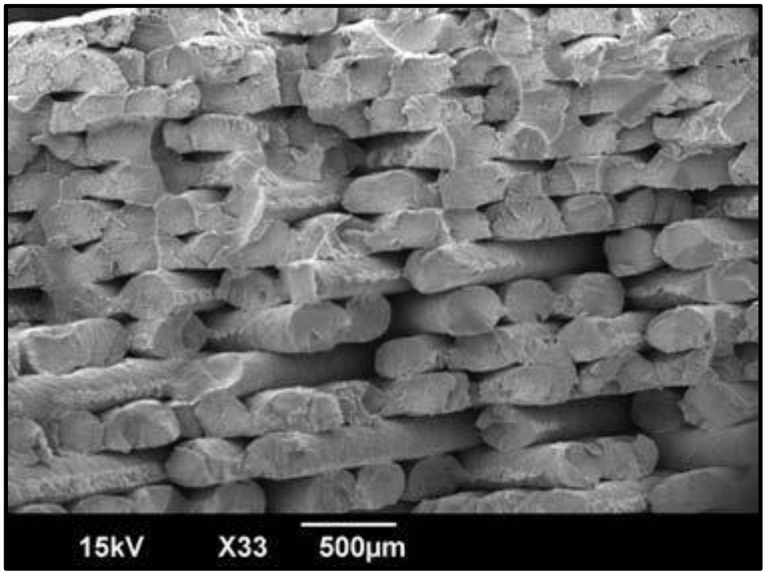


Figure 27 - Scanning Electron Microscope (SEM) Image of the Fracture Surface of a Multi-Layer Solid-Filled Part [24]

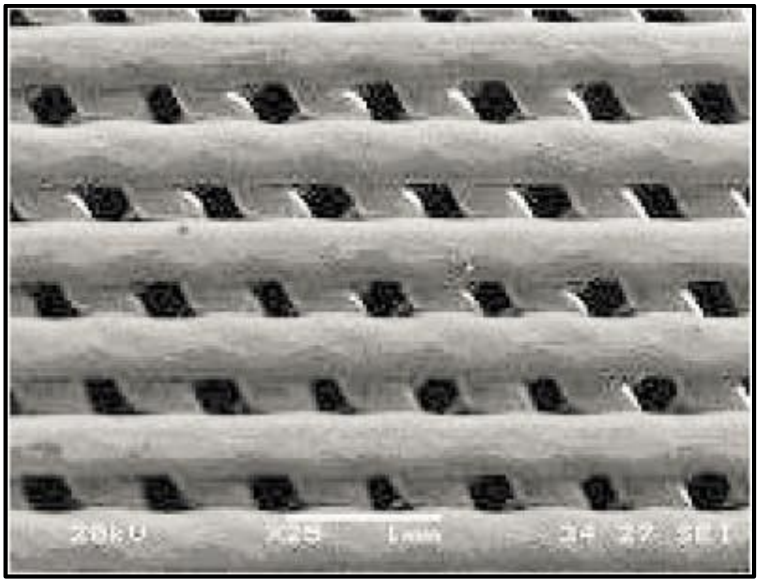


Figure 28 - SEM Image of a Three-Layer Sparse-Filled Part [25]

The effective cross-sectional area calculation was performed by calculating the mass fraction of the sparse and solid-printed models as compared to the weight of an injection-molded ABS test section (theoretically calculated using the density of P430 provided by Stratasys [26]). This approach was applied in the following manner:

$$A_{eff} = \frac{M_{FD}}{M_{IM}} * A_{Gage\ Section} \qquad \text{(Eqn. 3)}$$

The areas obtained for each specimen were applied to the engineering stress calculation, given according to the following equation:

$$\sigma = \frac{P}{A_{eff}} \qquad \text{(Eqn. 4)}$$

where P is the tensile force measured by the load cell and A_{eff} is the effective cross-sectional area of the specimen.

The effective area approach is similar to that employed by Rodriguez et al. during their FD-ABS materials testing. They presented the idea of using optical methods to measure the void density and applying the void density, ρ_1, to calculate the effective cross-sectional area of the specimen [27]:

$$A_{eff} = (1 - \rho_1) * A_{gage\ section} \qquad \text{(Eqn. 5)}$$

While this method would have been perfect for the sparse and solid-filled specimens' cross-sectional area calculations, Rodriguez et al did not provide explicit ρ_1 values for differing part densities. The previously-described mass fraction approach was therefore used instead.

3.3.1 MATLAB Script Functionality

Three MATLAB scripts were written specifically for the processing of the tensile test data. The first file, “Retrieve_Tensile_Test_Data.m”, imports all of the tensile test data in the Instron CSV files into the MATLAB environment. The script begins by prompting the user for the number of input files, followed by the file prefixes (defined by the user during file creation in the Instron Bluehill software package). It then creates a structure for each of the CSV files to store the following data: gauge length, width, thickness, extensometer displacement, extension rate, and load. The script loads these values as matrices of double values, then calculates the cross-sectional area from the measurement for thickness and width (measured with calipers just before testing). The areas can then be manually adjusted according to Equation 3. Strain is calculated from the extensometer displacement and the gauge length, while the stress is calculated from the effective cross-sectional area and the measured load.

With these calculations completed, the stress and strain data for each sample is stored in separate matrices within the file structure. Each structure is manually saved by the user by right-clicking the structure and saving the data in MAT format. The next step is to run the "Plot_Tensile_Test_Data.m" script. This script loads the .MAT file for a specific sample of data (the script must be run once per attack sample). It then plots the stress and strain matrices for each test sample with individual data series for each specimen within that sample.

Since the ultimate tensile strength provided a good metric of the worthiness of each attack, the statistical mean and deviation for each sample were also calculated using a specially-tailored MATLAB script. This script, "Statistical_Calculations_for_Tension_Testing.m", loads all of the MAT files for every attack sample. It then calculates the mean and standard deviation of the ultimate tensile strength for each specimen.

3.4 Tensile Test Results

The results including the fracture properties from the tensile for each attack were compared against the control case and the results were informative. The specimen chosen for this control case was used in the Stratasys-published material testing of ABS-M30, the Solid-Filled On-Edge (XZ) orientation [19]. The fracture behavior of this control sample is illustrated below:

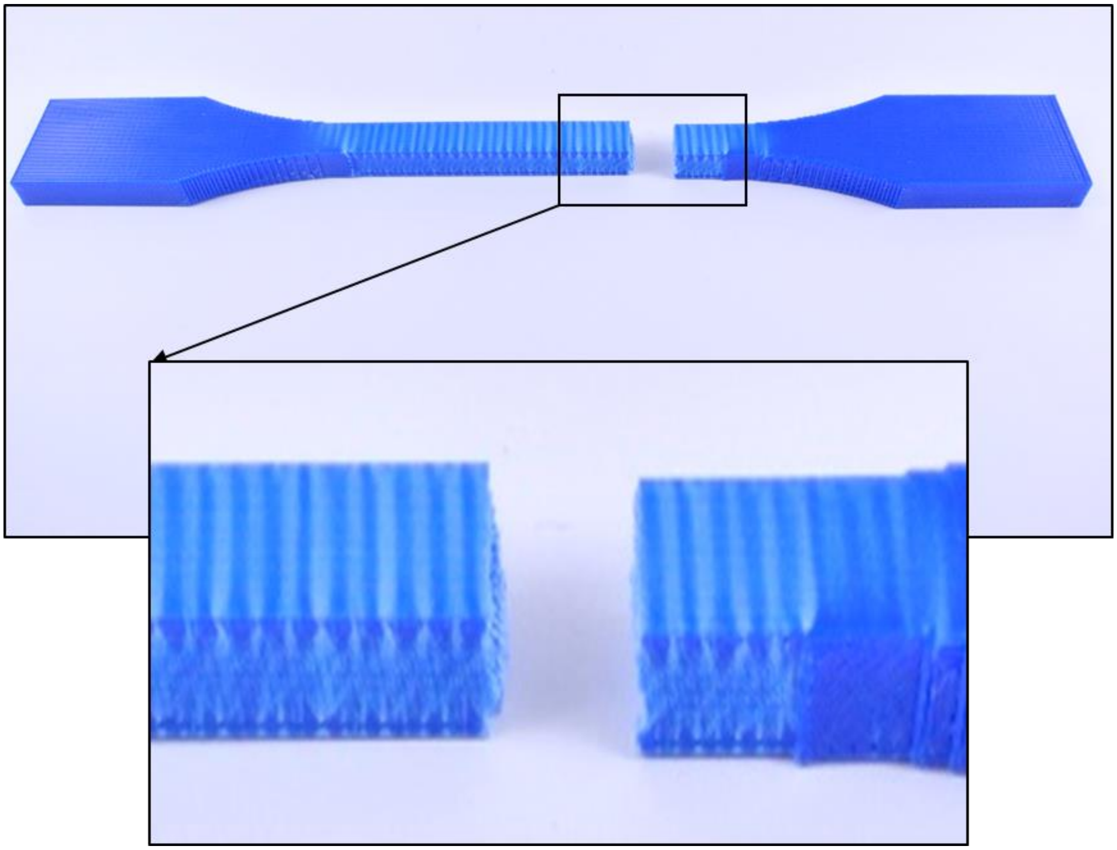


Figure 29 - Broken Solid-Filled On-Edge Specimen (Control)

The stress versus strain plot for this control specimen is shown below:

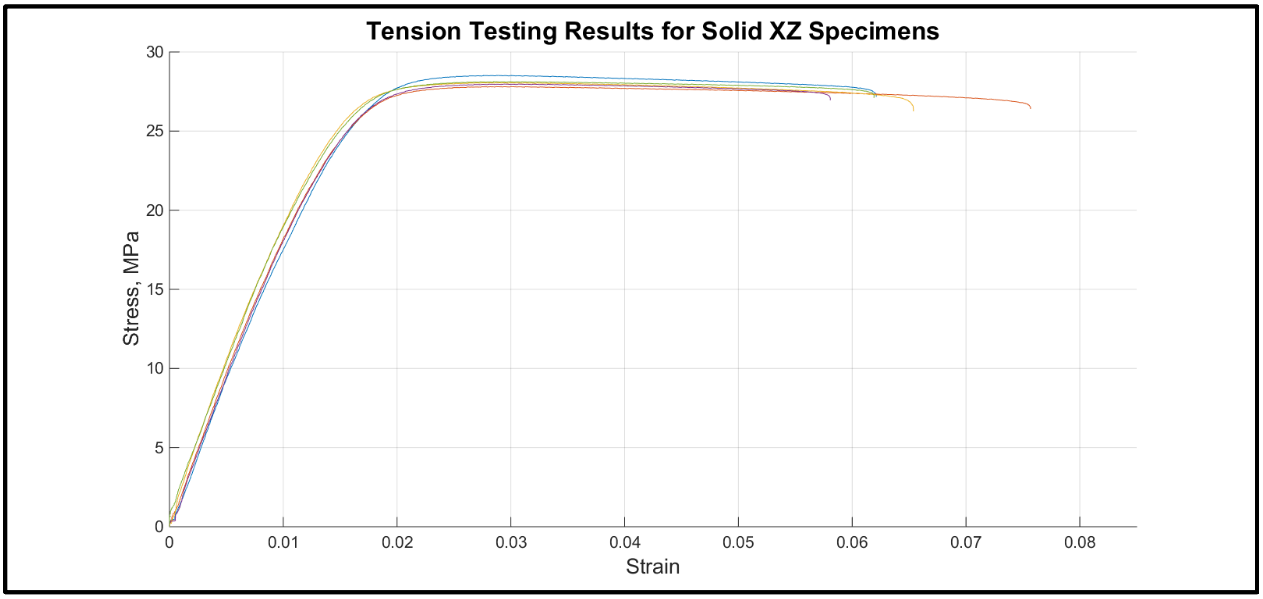


Figure 30 - Control Specimen Tension Testing Results

3.4.1 Support Material Swap Attack

Part material in the interior fill was replaced with support material, a much more brittle thermoplastic that dissolves in a sodium hydroxide bath. The danger of this attack stems from the fact that the total material usage remained the same and the apparent weight of the finished part was roughly identical to that of the control specimen. External appearances were also unchanged. Part strength was drastically reduced since these two polymers did not bond thoroughly to each other, causing rapid delamination during loading and resulting in brittle failure behavior. The test results indicated a drastic reduction in strength to failure.

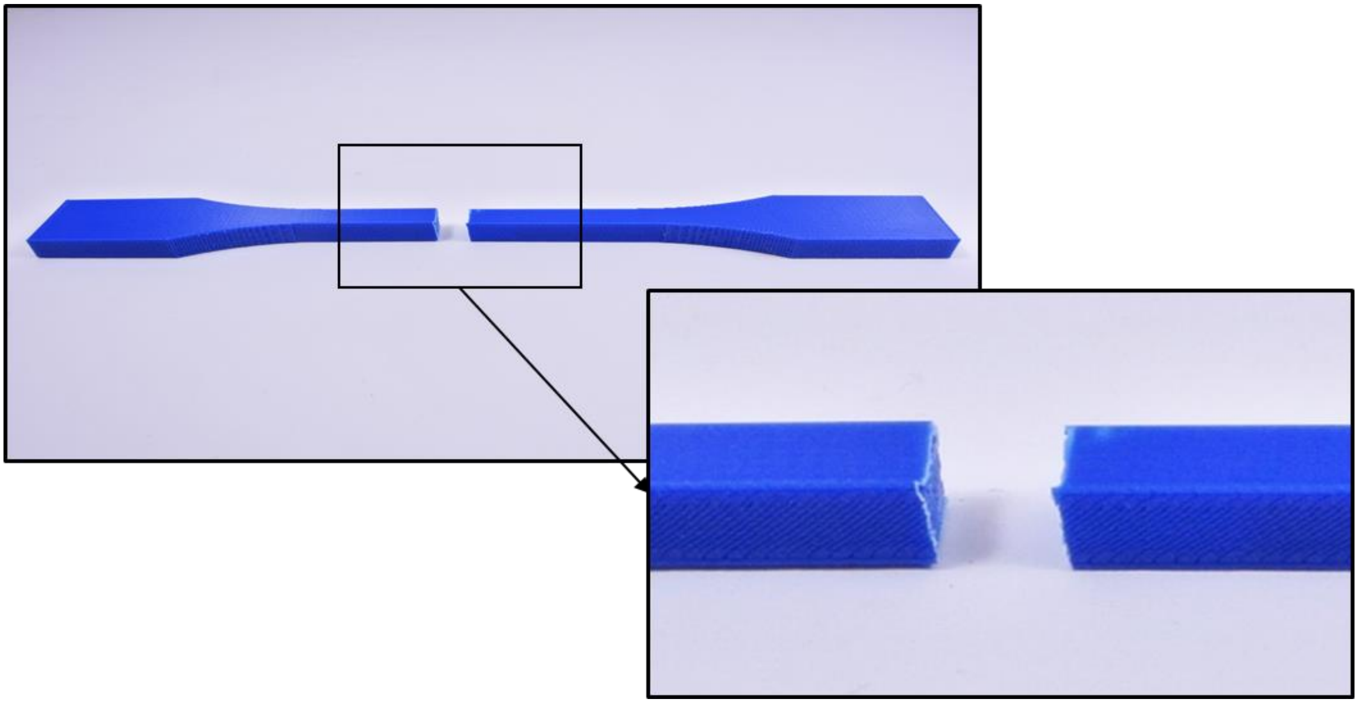


Figure 31 - Attack No. 1 - Broken On-Edge (XZ) Specimen Filled with Support Material in the Gauge Section

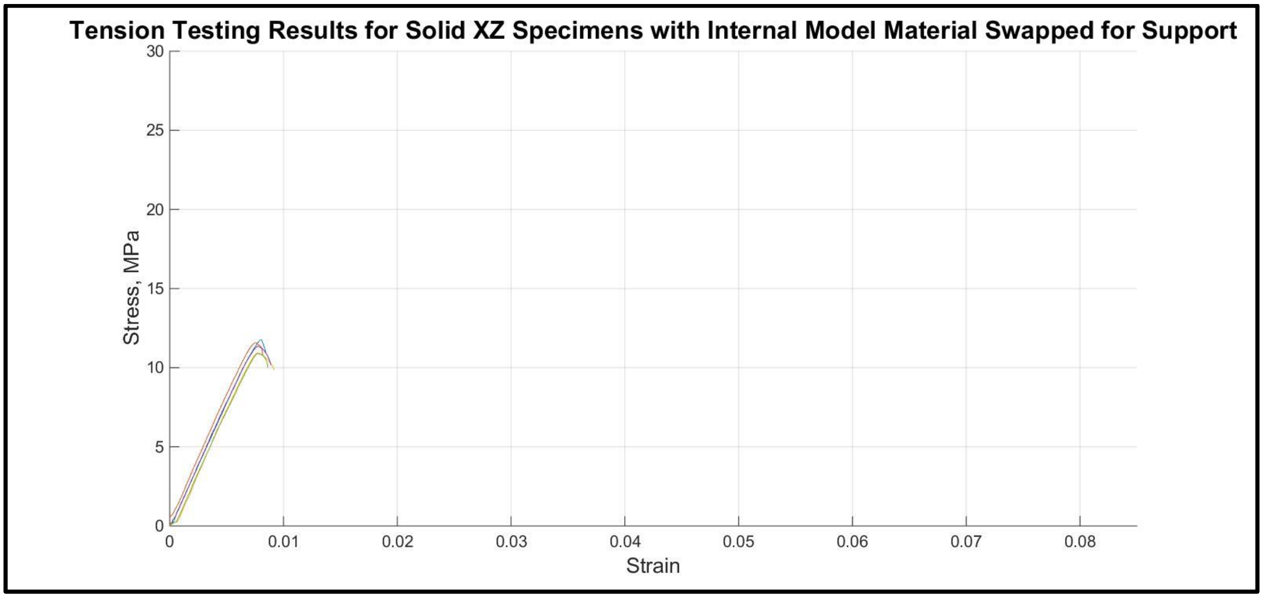


Figure 32 - Attack No. 1 – Tensile Test Results for Broken On-Edge (XZ) Specimen Filled with Support Material in the Gauge Section

3.4.2 Fill Density Adjustment Attack

The second attack involved adjusting the fill density of the test specimen. Initially, the internal fill density of the entire specimen was changed from a solid in-fill pattern to a sparse, high-density (SHD) in-fill pattern. However, the density reduction in the grip section induced fracture outside of the gauge length for this specimen, invalidating the results. This attack method was also difficult to execute and easily detectable through weight estimates. The attack was therefore reformulated to produce a localized density reduction in a small area of the gauge length through the manipulation of the road width parameters. With density reduction confined to a small section in the gauge length, stress concentrations were no longer induced into the sections outside the gauge length and fracture occurred in the gauge section.

The reduction in toolpath road width reduced the contact area and mixing between adjacent filaments, further reducing the bond strength. The filaments were therefore in the correct location, but only weakly bonded together. The result was a failure method with a premature yield strength as well as a low ultimate tensile strength:

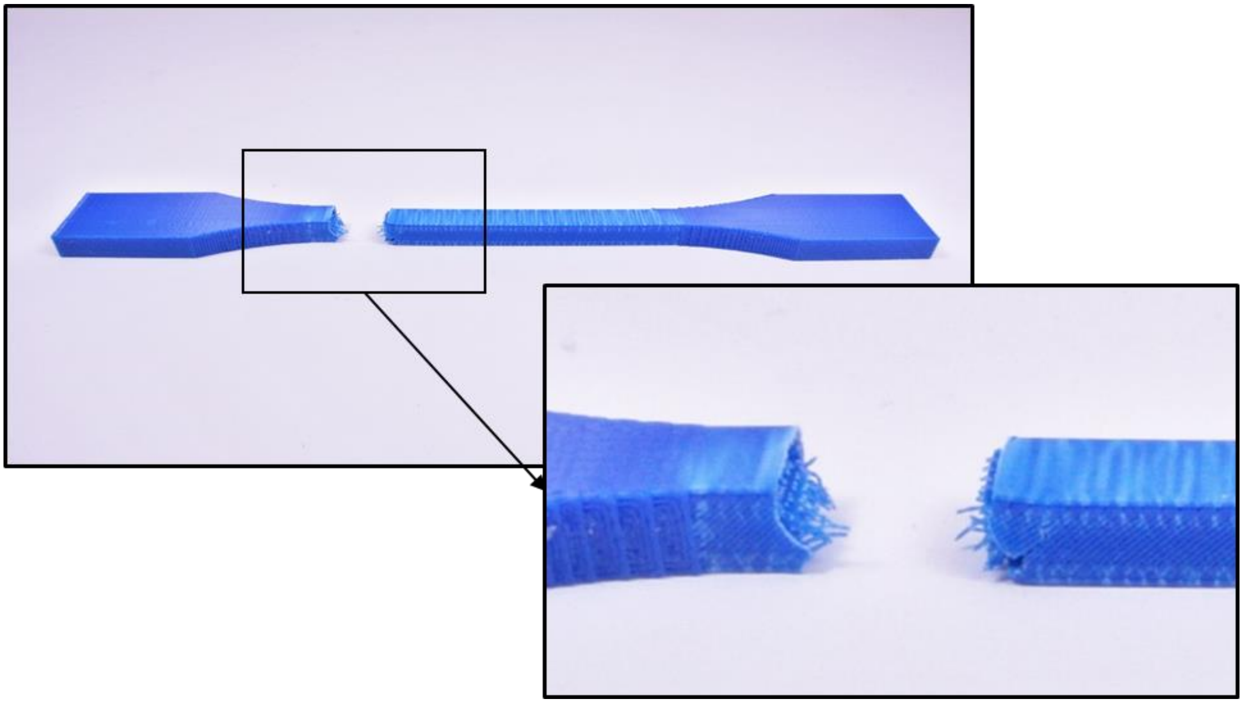


Figure 33 - Attack No. 2 - Broken On-Edge (XZ) Specimen with Fill Density Alteration

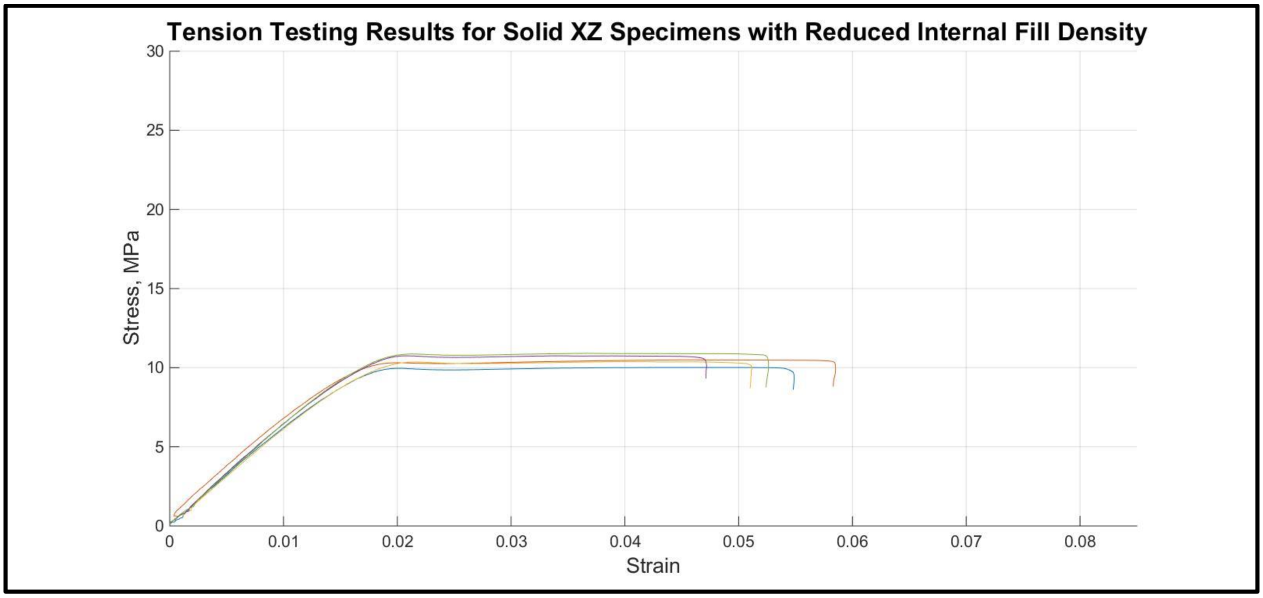


Figure 34 - Attack No. 2 – Tensile Test Results for Broken On-Edge (XZ) Specimen with Fill Density Alteration

3.4.3 Notch Insertion Attack

Valid tensile test results were obtained from testing the specimens that had cracks deliberately added into the gauge section. This condition caused crack propagation at a specified location, remaining within the gauge section. The behavior of this specimen is interesting since the material removed from the part was unnoticeably small (only 0.001% of the total model volume). A detailed image of the fracture is shown below:

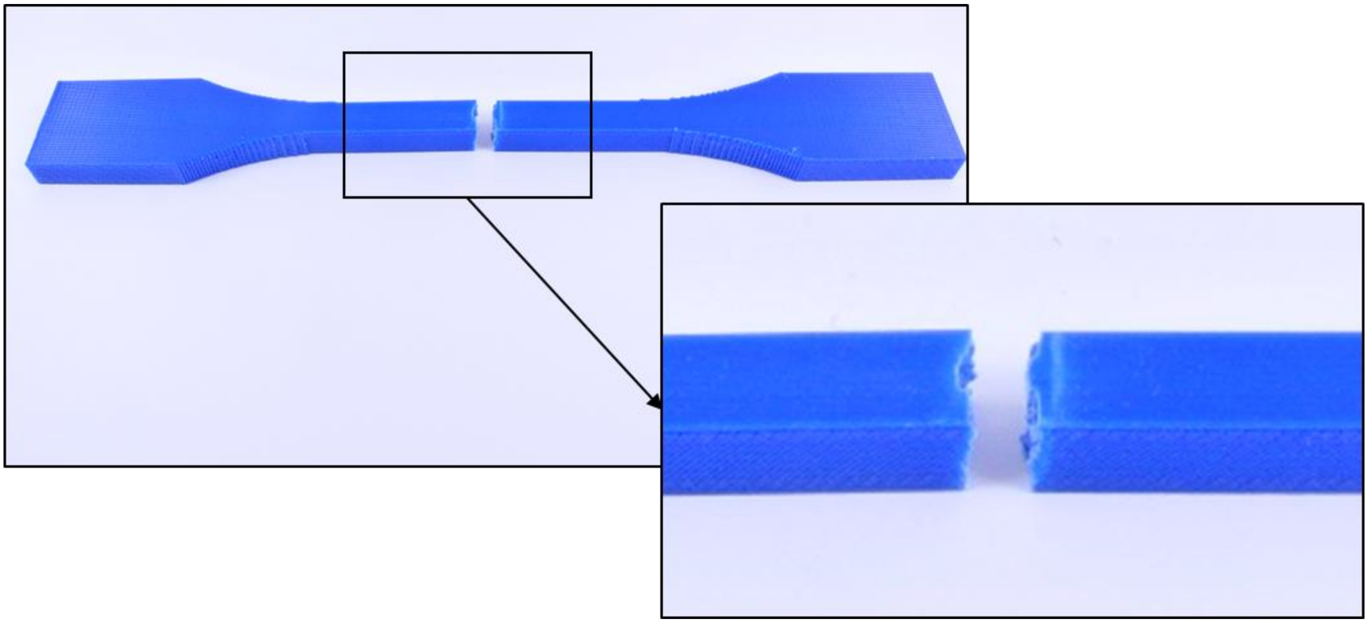


Figure 35 - Attack No. 3 - Broken Solid-Filled On-Edge (XZ) with Notch in Center of Gauge Section

The image above shows that this specimen underwent a clean break with hardly any strain outside of the fracture zone. The dark blue plastic in the gauge section shows a visual confirmation that this part underwent very little strain before fracture. The attack caused a brittle failure mode for this part, whereas the control sample exhibited a much more ductile response. The fracture behavior is easily seen in the following stress versus strain plot:

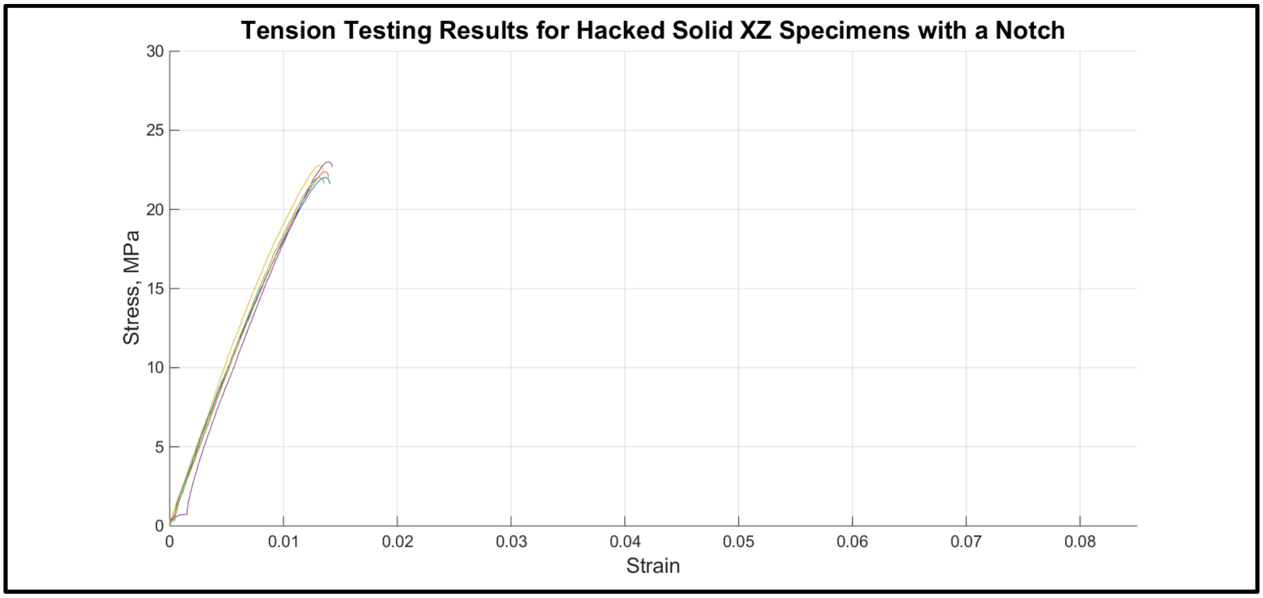


Figure 36 - Third Attack Tension Testing Results

This plot shows that not only does the part fracture in a much more brittle manner, but the part's ultimate tensile strength is reduced due to the weak bond created by the wider gap between adjacent filaments. Please note that this the *apparent stress* vs actual strain for this specimen. For true stress, the stress calculation would have to be adjusted for the lack of a proper bonding surface. However, the weak or missing surface bonds represent the most dangerous aspects of this attack.

3.4.4 Seam Manipulation Attack

This attack moved the part seam from the end of the specimen (as found in the control specimen) to the center of the gauge section. This placement was chosen to simulate the movement of a part seam to an area of localized high stress. The majority of these samples broke in a predictable manner right on the seam. However, one of these five specimens broke slightly away from the seam at an unanticipated printing defect. This demonstrated that even though the seam represented a printing defect, other naturally-occurring defects due to loss of extrusion (perhaps due to a temporary tip clog) or similar errors could also have caused premature failure. The following image shows the fracture surface:

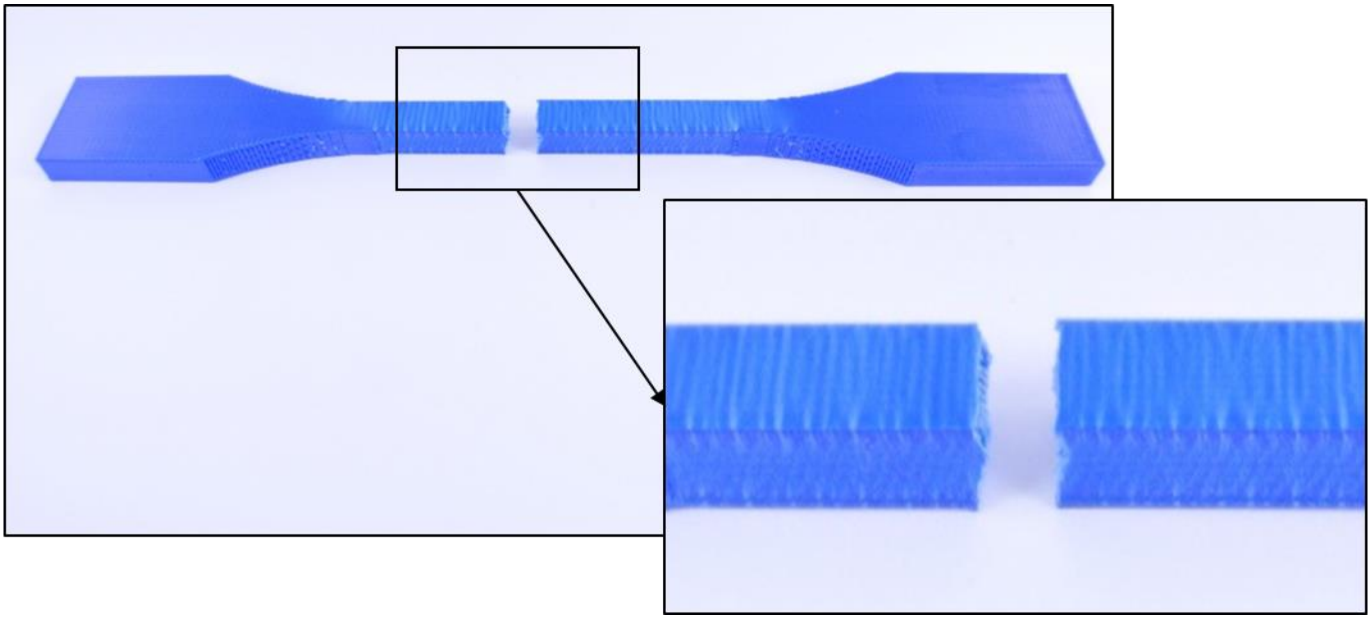


Figure 37 - Broken Solid-Filled On-Edge (XZ) with Seam in Center of Gauge Section

The weak bonding at the seam caused the first crack to form there. As previously stated, polymer bonds at locations with discontinuous extrusions are thermally-induced, sintered bonds [21]. Insufficient heat flux or variations in polymer alignment make these bonds unpredictable. The results from the tensile tests of these specimens are shown below:

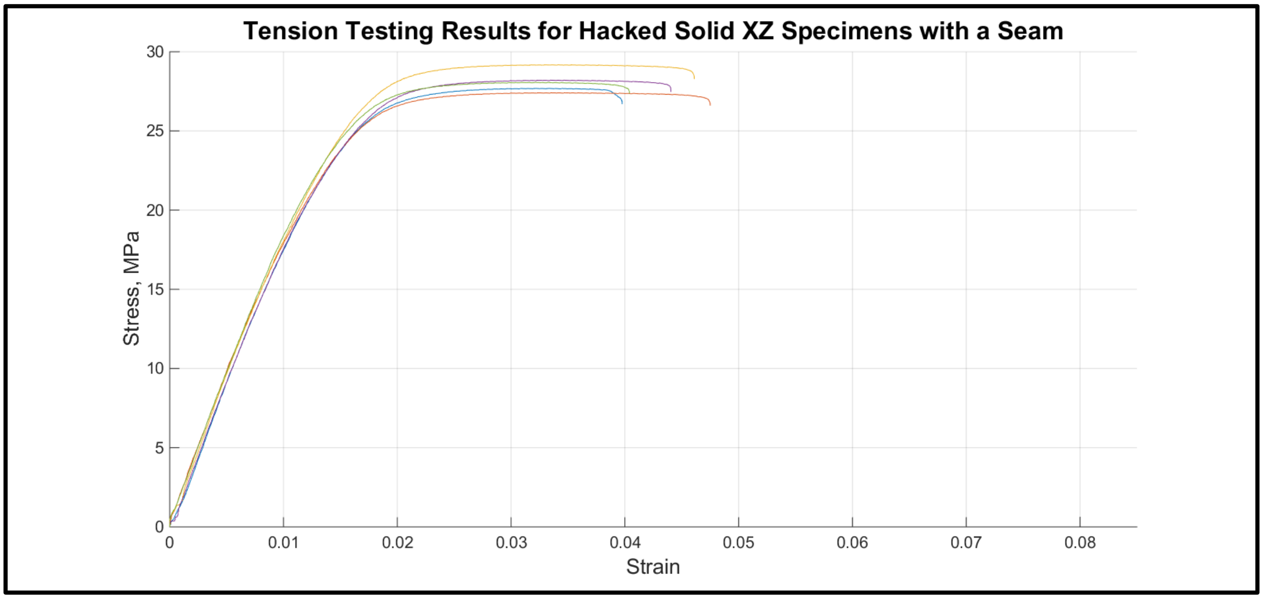


Figure 38 - Fourth Attack Tension Testing Results

It is clear from these plots that placement of the seam in a high-stress location led to brittle fracture behavior, with the model experiencing a lower average strain to failure. The specimens with the shortest strain to failure from the control case displayed similar strain values as the specimen with the longest strain to failure from the attack case. This is due to the random printing defects introduced in the control specimens during the printer's normal operation. These uncontrollable defects can be weaker than the seam defects in cases of improper alignment of the seam bonds.

3.4.5 Temperature Set Point Alteration Attack

The final attack prescribed interference of the printer's extruder head temperature. The extruder head temperature was altered around the original set point by 20°C in both directions, in increments of 10°C. This resulted in five batches of samples to test, with each batch composed of five specimens. This number of test specimens was chosen to provide a sufficiently large sample size to include any printing variations. To control printing variances, the test samples were all obtained from the same printing pack (they were built at the same time with the same machine settings on the same build plate). To find the default printing temperatures, control samples were printed with the RTD temperature monitoring system in place on the model extruder. Tip temperature data was logged throughout the build process, and the temperature variations for the process are shown in the plot below:

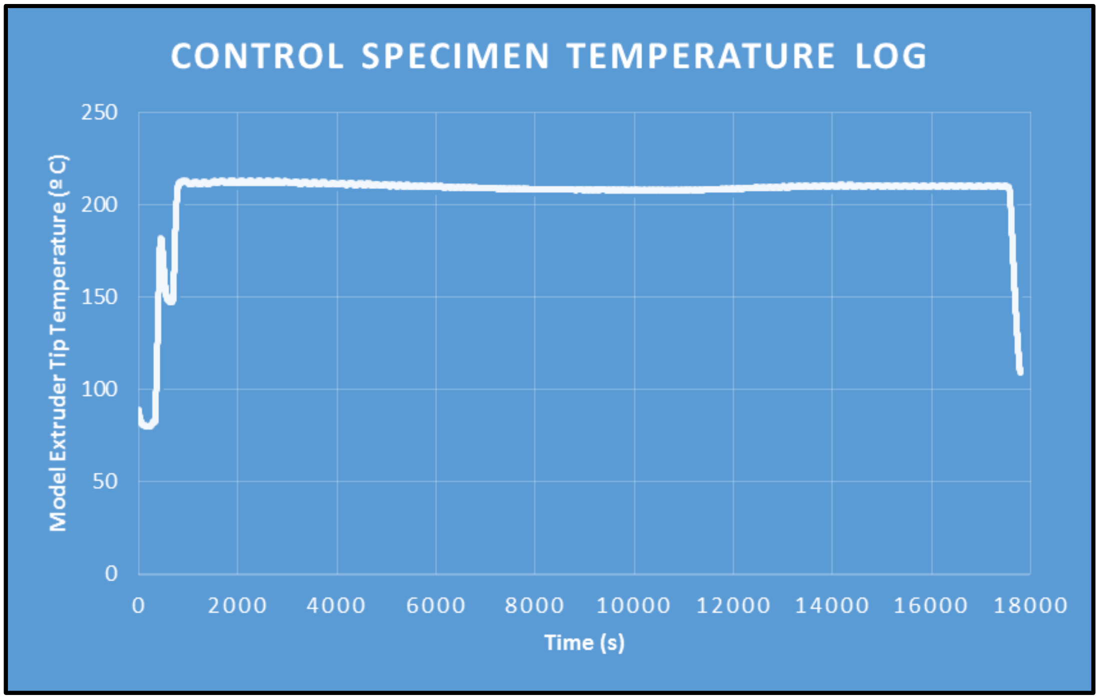


Figure 39 - Temperature Log for Control Specimen Print

The temperature sensed by the RTD monitor at the extruder tip was 210°C throughout the build process. The temperature adjustment drastically affected the tensile properties of the specimens, as shown in the stress versus strain plots on the following page:

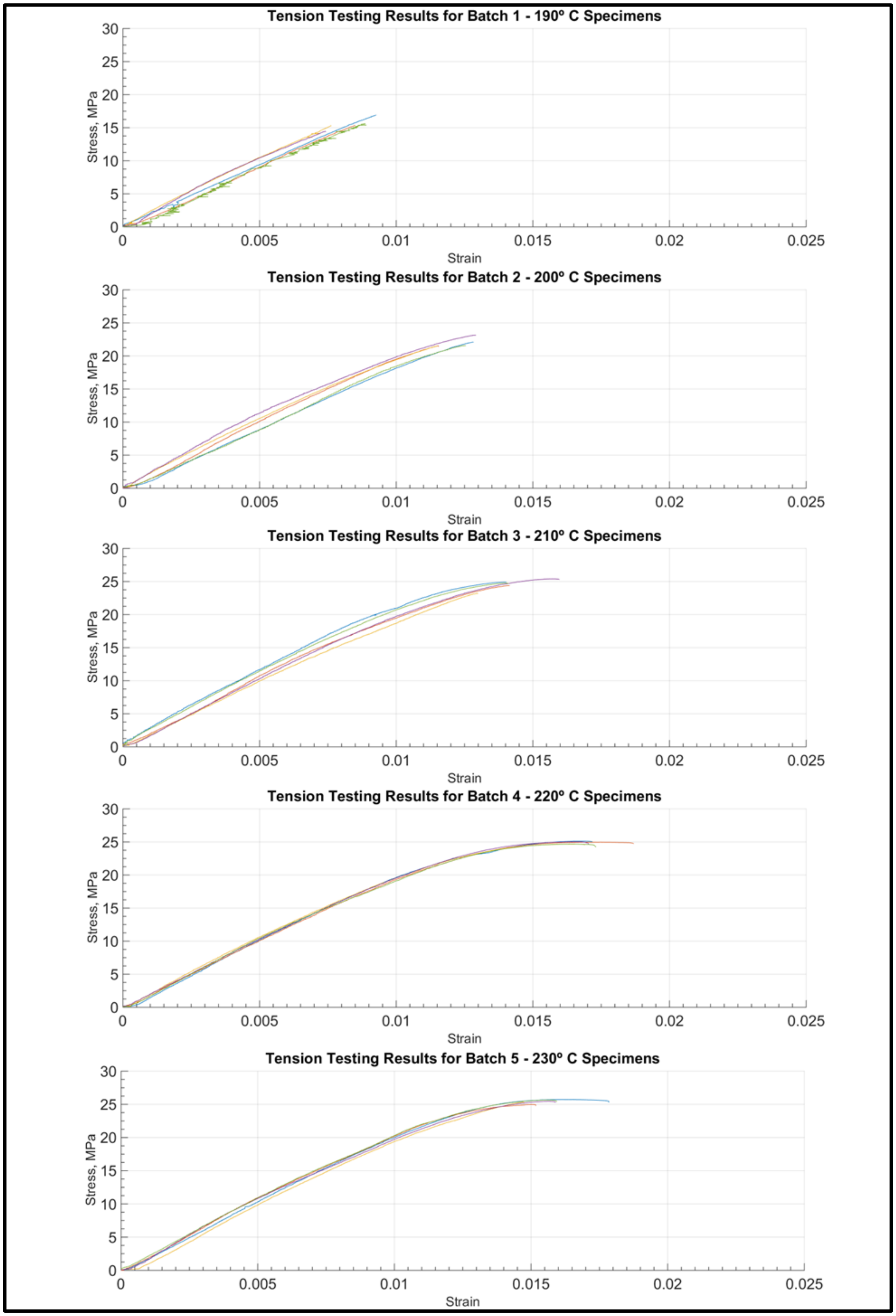


Figure 40 - Tensile Test Results for Temperature Attack

The first plot shows that reduction of the tip temperature by 20º C during printing can reduce the ultimate tensile strength by approximately 37%. The weaker lamination is evident in the following image of the fracture surfaces of the altered specimens, as compared with the control specimen (210º C):

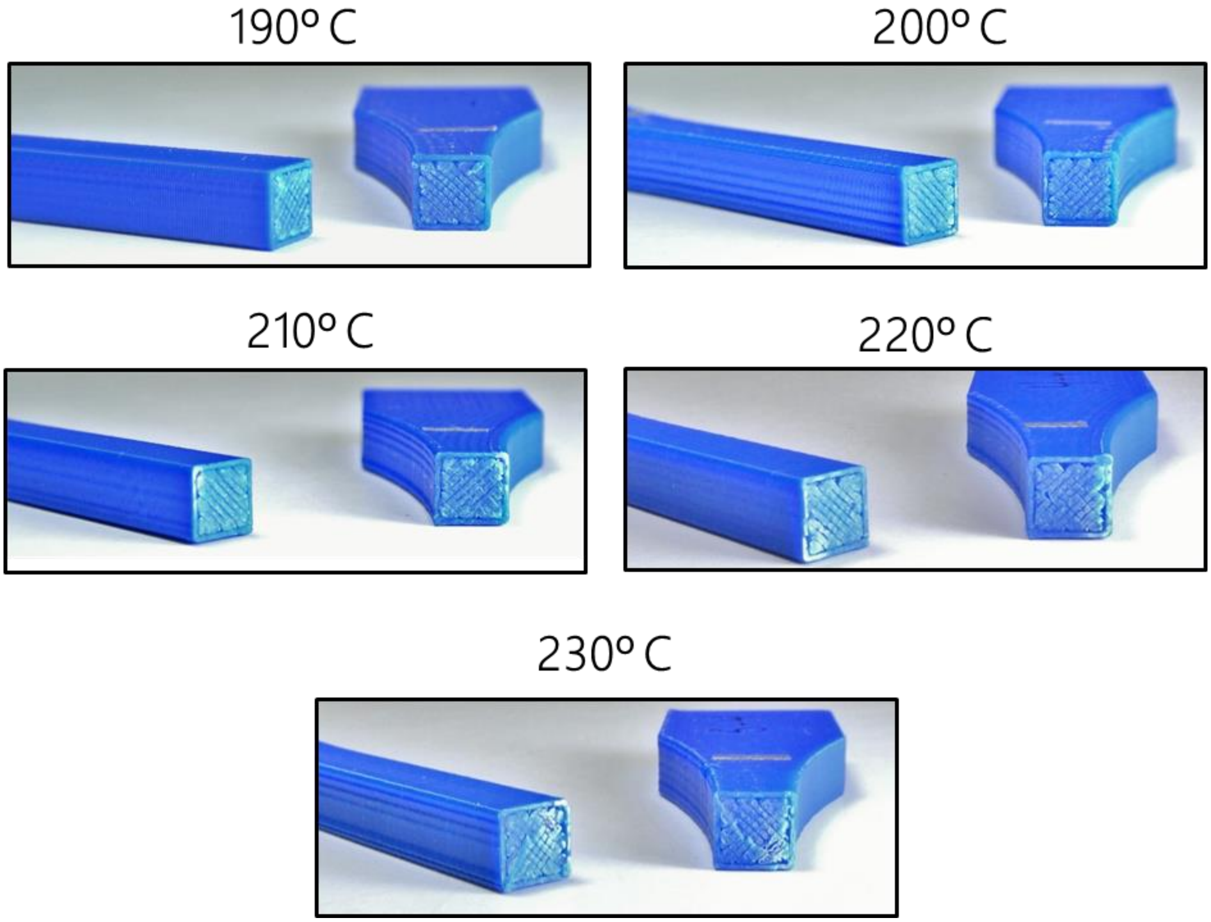


Figure 41 - Fracture Surfaces of Temperature Attack Specimens

The strongest inter-layer bond was seen in the 230°C specimens. The bonded layers sintered properly, such that the part tore between layers instead of delaminating. The temperature was still low enough, though, that the increased tip temperature did not yet thermally degrade the polymer extrusion.

Chapter 4 - Design of a Secure System Monitor

4.1 Security Concerns for 3D Printers

The aforementioned tests demonstrate the feasibility of attacks on additive manufacturing systems. More sophisticated industrial sabotage is possible with the recent inclusion of internet connectivity in manufacturing equipment. Models created by additive manufacturing equipment are becoming increasingly attractive for use in non-industrial environments, as mass-produced items can now maintain some aspect of uniqueness. The current poster child for this mass-customization movement is Invisalign, a company that manufactures orthodontic braces. Invisalign prints plastic dental “aligners” from 3D models of the patient’s mouth (obtained from X-ray scans), thus creating custom-fit orthodontic appliances at a mass-production level [28]. Military branches have been also using 3D printers to make custom parts at forward operating bases with poor supply chain access in hopes of establishing procedures for repairing equipment [29]. This diverse range of 3D printer usage provides cyber attackers with a physical consequences for malicious attacks.

Current defensive security techniques rely on the security of the printer’s primary network, such as the Local Area Network (LAN) that the printer inhabits. Depending on the level of access, this could also include the security of the servers the printer accesses over the internet. When threats are detected on the primary network, the firewall software should be patched to allow early detection of such attacks. Barrier security methods can provide sufficient protection against the transfer of files that have been intercepted and infected via unauthorized network access. The System Aware Cyber Security solution proposed in this research instead considers the scenario in which the file is not corrupted on the network, but rather on the machine itself:

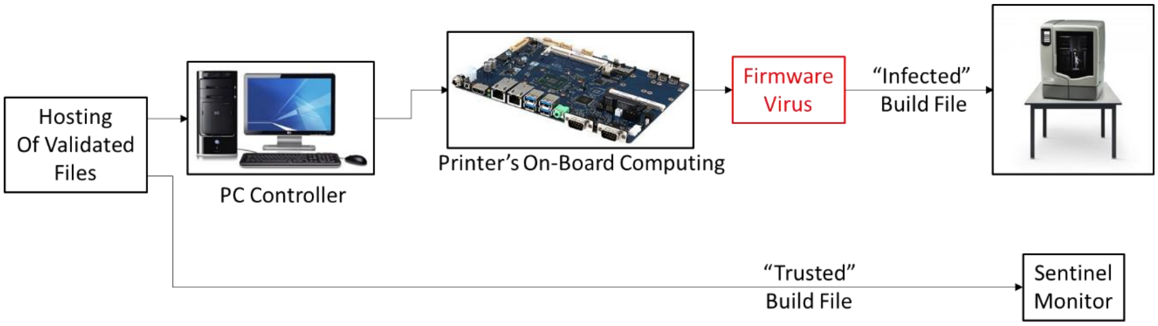


Figure 42 - The Flow of Information in a Sentinel-Monitored Printer with a Firmware Virus

A malicious attacker with either physical access to the machine or brief network access to the machine's PC controller could corrupt important hardware or software. Many of these machines have the ability to update the printer firmware with similar methods of access. The attacker would only need brief network access, since they could rewrite the printer's ROM and then terminate the connection once the file transfer had been completed. A physical alteration of the firmware (someone swapping a hard disk or memory chip) might occur in a manner that was undetectable to the network security software. For this reason, a new approach to securing mechatronic systems is proposed.

By definition, mechatronics is the "synergistic integration of mechanical engineering with electronics and intelligent computer control in the design and manufacture of industrial products and processes" [30]. The computer controller on a 3D printer receives an input signal and transforms it into a physical output motion. For electro-mechanical actuators, this input signal is either an analog or digital voltage signal. Some property of the voltage signal, whether it is the amplitude, frequency, phase, etc., dictates the mechanical output motion from the actuator. This motion-based output outlines the fundamental difference when securing mechatronic devices: incorrect processing of the input data stream leads to faulty motion parameters. Addition of a monitoring system to track these motion parameters therefore becomes necessary in order to guarantee the integrity of parts created by 3D printing processes. The proposed solution is a System-Aware Cyber Security monitoring device (called a sentinel) that tracks, logs, and predicts the behavior of the 3D printing process. This sentinel device is essentially a smart data logger that has access to key operational parameters. This allows the sentinel to compare the data it collects from machine sensors against data from a validated set of build parameters.

4.2 Secured Parameters for uPrint Application

In its current design, the sentinel monitor must be custom-fit to the printer for each application. There is not currently a universal sentinel monitor; each mechatronic process in need of protection contains unique data streams and key parameters, and the sentinel device must therefore be custom tailored for each application. The varying printing mechanisms and build parameters require different monitoring equipment for the sentinel, affecting both the hardware and software interface for the sensors and controllers. For example, while a sentinel monitor for an FDM printer might use an encoder to measure the extruder's rotational speed, the sentinel monitor for an SLA printer would not monitor this parameter and therefore would not require the installation of an extra sensor. Even if some parameters among different printer technologies are similar, they may require different types of sensors for measurement purposes. A printer utilizing FDM technology was chosen for this case study, so the sentinel was tailored to include the necessary sensors and monitoring equipment required for this technology. Some redundant sensors were fitted to the machine to allow for operation independent from the monitored machine, while other feedback data was simply probed (such as data from existing sensors). The five major attack vectors provided in Chapter 3 affect the following operational parameters: Print

Material, Extrusion Rate, Height Offset, Extruder Tip Temperature, and XYZ Position. The values for each of these print parameters is determined in different ways by measuring data output streams from certain sensors. These are described in the following sections.

4.2.1 Print Material

The current print material was determined in the Stratasys Dimension uPrint through measurement of the status of the toggle head sensor. The filament drive mechanism contains a single motor that drives both filaments, as shown in the following figure:

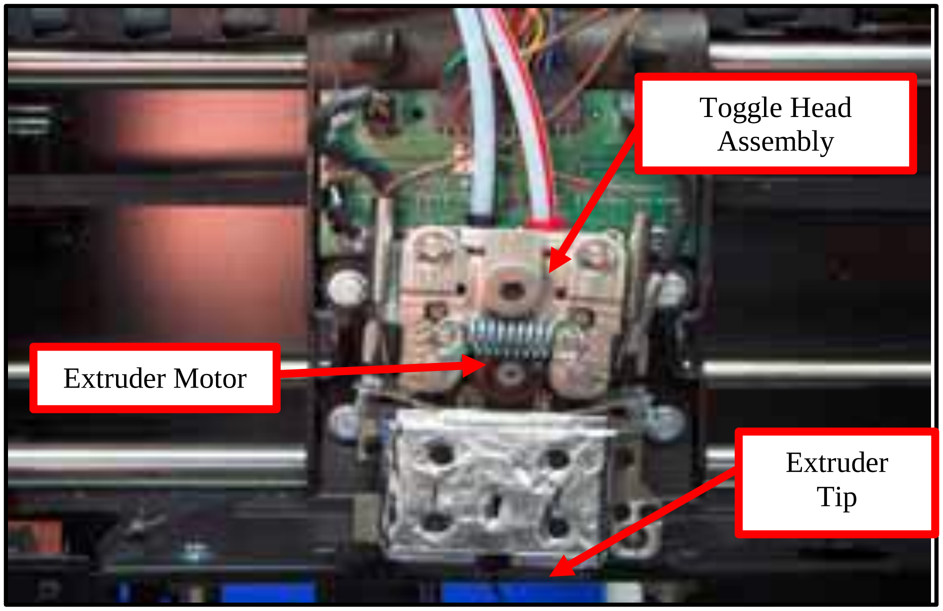


Figure 43 - Extrusion Motor Drive Mechanism [18]

During extrusion, the model material filament enters through the red tube (shown above), and into the toggle head assembly. This mechanism rocks back and forth to push either the support or part material filament against the drive motor. The gantry head fitted to this rocker mechanism has a mechanical switch that flips during the head position toggle maneuver. The output of this switch can be used to detect the current print material. To detect the extrusion rate, extruder motor encoder feedback must be provided to the sentinel device (see Section 4.1.2). However, this method for print material or extrusion rate detection was unreliable, since the motor changed rotational directions and speeds frequently during an individual toolpath between two specified coordinates. The extruder back-drives the filament before it reaches the last waypoint in a toolpath to reduce superfluous extrusion caused by the die swell effect, which is a characteristic expansion of filament as it passes through a convergent nozzle such as the extruder nozzle [31]. This phenomenon causes the plastic filament to continue extruding despite a lack of applied pressure. This unwanted extrusion is called drool, and it is one of the reasons that all FDM parts contain a visible seam.

Because of this reverse rotation at the end of the toolpath, the best method for detecting current material usage was splicing and monitoring the signal from the toggle head sensor. The toggle sensor output was therefore provided to the sentinel monitor, where a sentinel input channel was dedicated to monitoring the print material signal.

4.2.2 Extrusion Rate

As previously mentioned, the extrusion rate can be retrieved from the encoder signal from the extruder motor. The extruder motor in the uPrint is a precision Maxxon gear-motor fitted with a high-quality quadrature encoder. The inclusion of an encoder in this system allows for closed-loop control of the extrusion speed, providing both direction and speed data to the controller. In order to interpret this signal, however, it is important to know the structure of encoder signals.

There are two main types of encoders: incremental and absolute. An incremental encoder only provides a signal when the motor's position changes. A trigger device moves past a sensor inside the encoder housing, causing a pulse train in the output signal as illustrated in the right-hand side of the following figure:

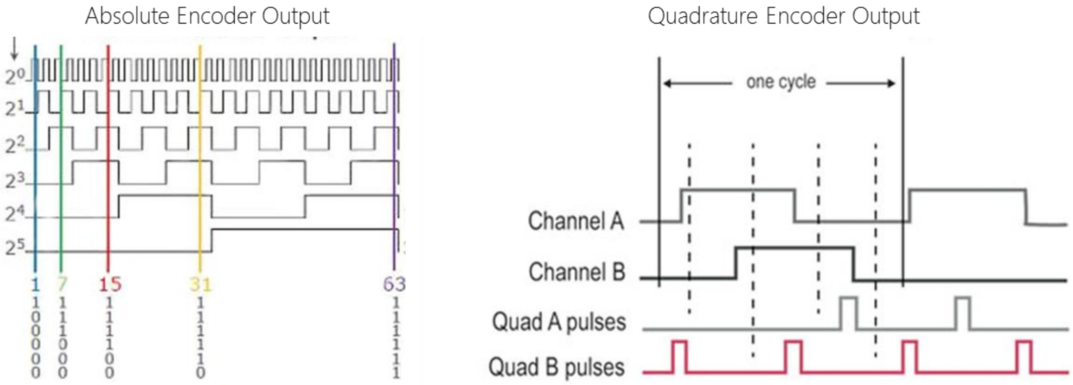


Figure 44 - The Difference between Absolute and Quadrature Encoder Signals [32]

An absolute encoder, however, transmits a number that corresponds to the angle of the encoder shaft. The absolute encoder signal is usually formatted according to some serial transfer protocol. Absolute encoders have a distinct advantage over incremental encoders: absolute encoders will not lose track of their overall position if they skip steps, making absolute encoders better in applications for measuring precise angles where the machine might move faster than the encoder can output data. However, when using an encoder to determine angular speed, incremental and absolute encoders work equally well. The incremental encoder implemented to measure the angular rotation of the extruder head in these experiments was quadrature, producing two channels with pulse trains 90˚ out of phase (see Figure 44). The staggering of the output channels (which signal leads or lags the other) indicates the rotational direction of the shaft. The source channel of the leading output pulse indicates which direction the motor is spinning. Thus, the output of the extruder motor's quadrature encoder contains both the rotational direction and the rotational speed of the encoder. The rotational speed of the extruder motor is also known as the Extrusion Rate. It can be calculated from the time between successive encoder counts according to the equation below:

$$ER = \frac{(D_{roller})(R_{fd})}{(t_{cnt})(CPR_{enc})} \qquad \text{(Eqn. 6)}$$

Where ER is the Extrusion Rate, D_{roller} is the diameter of the extruder's toothed roller, R_{fd} is the final drive ratio of the gear transmission between the drive motor and the output gear, t_{cnt} is the time between successive encoder counts, and CPR_{enc} is the number of counts per revolution of the encoder. The Extrusion Rate also demonstrates when the extruder turns off. If ER drops to 0, then the motor is not spinning and no material is extruding.

An initial method for extrusion detection calculated the extrusion rate in real time, but rapid fluctuations in the extrusion rate as it followed the onboard motion control chip's interpretation of the CMB file's movement commands ultimately proved this method to be impractical. A new method was developed to monitor the rotational status of the extruder motor, but this proved problematic in the final testing stage. For final extrusion detection, a material usage approach was applied, which recorded the total distance traveled by the extruder motor. This method proved to be much more reliable, but it masked the overall detail in the extrusion data. The reasoning for this alteration is detailed in Section 5.2.

4.2.3 Height Offset

The Height Offset parameter controls the road width of the printed filament as illustrated in the figure below:

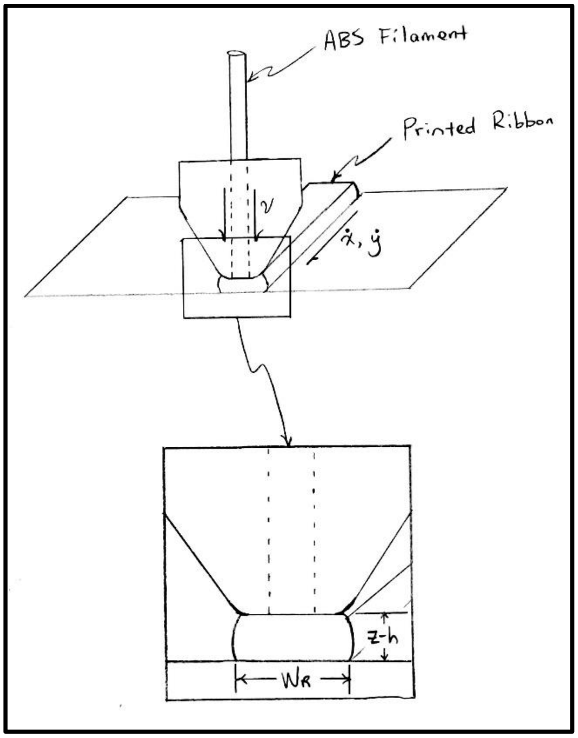


Figure 45 - Diagram of an Extruded Road

ABS filament is extruded through the liquefier nozzle at the extrusion rate v, while the extruder's gantry head travels at translational speeds $\dot{x}$ and $\dot{y}$. The current Z axis position is denoted as z, so the road width W_R is directly controlled by the height offset h and the extrusion and travel speeds. The nozzle smooths the top layer of the filament as it runs, laying down a uniform ribbon on the build tray. When the height offset is increased, the filament is more severely deformed, increasing the road width. The road width can also be affected by variances in the travel and extrusion speeds, as shown in the picture below:

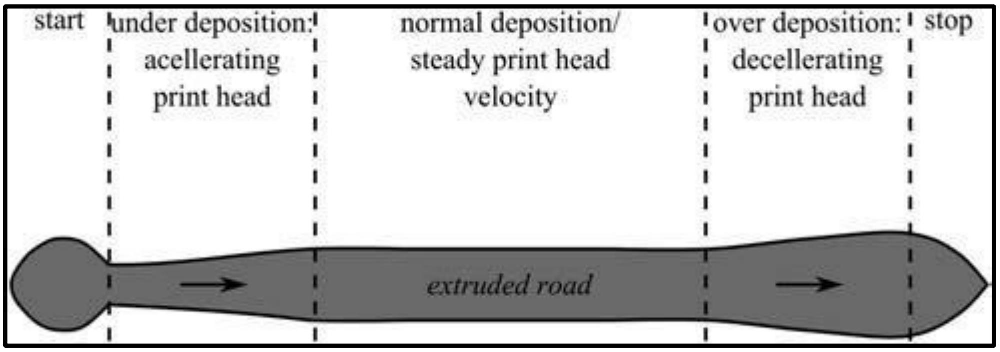

start
under deposition: acellerating print head
normal deposition/ steady print head velocity
over deposition: decellerating print head
stop
extruded road

Figure 46 - The Effect of Under- and Over-Deposition (Wrong height offset or travel speed) [33]

The travel speed for the uPrint is controlled by its move compiler. This on-board chip interprets the parameters fed into the printer in the build file and calculates the proper travel speeds and extrusion rate from the road width. The travel speeds could be obtained from the incoming data stream from each encoder. It was discovered, however, that road width is controlled solely through the extrusion rate, as the height offset parameters for the uPrint only change with the current print material. The height offset is therefore only used to adjust each waypoint's Z location based on the current print material.

4.2.4 Extruder Tip Temperature

The Extruder Tip Temperature is perhaps the most important print parameter for an FDM printer. As mentioned earlier, ABS plastic is amorphous, meaning that there is no real melting temperature. Once the glass transition temperature is reached, the material starts the flow, but not without high shear stresses in the flow field. In fact, the fluid shear stress in the liquefier is strongly dependent on the fluid viscosity, which is a function of temperature [22]. To reduce these shear stresses, the temperature must be increased, but increasing the tip temperature too far can also have detrimental effects. Not only can polymer chain interactions suffer, but it has also been found that certain compositions of ABS plastics, when thermally degraded, form toxic hydrogen cyanide gas [9, 34]. It is therefore critical to keep the temperature in a range below the boiling point of the filament and above the glass transition temperature for significant reduction in fluid shear forces. The possibility for the emission of cyanide gas from normal ABS hints at chemical attack vectors for enclosed operational environments through which normal or deliberately altered materials could be made to release dangerous substances into the air that might damage equipment and injure nearby personnel. Tip temperatures therefore require constant supervision to ensure safe operation of printing equipment.

A Dimension uPrint with OEM material regulates the tip temperature to 210-215º C during a typical build process. The sentinel devices utilized a specially-designed temperature measurement system with a platinum resistive temperature detector (RTD) sensor implanted in the printer's tip shields to measure the tip temperature directly. The sensor was bonded to the extruder tip with a small amount of thermal grease for reduced thermal contact resistance.

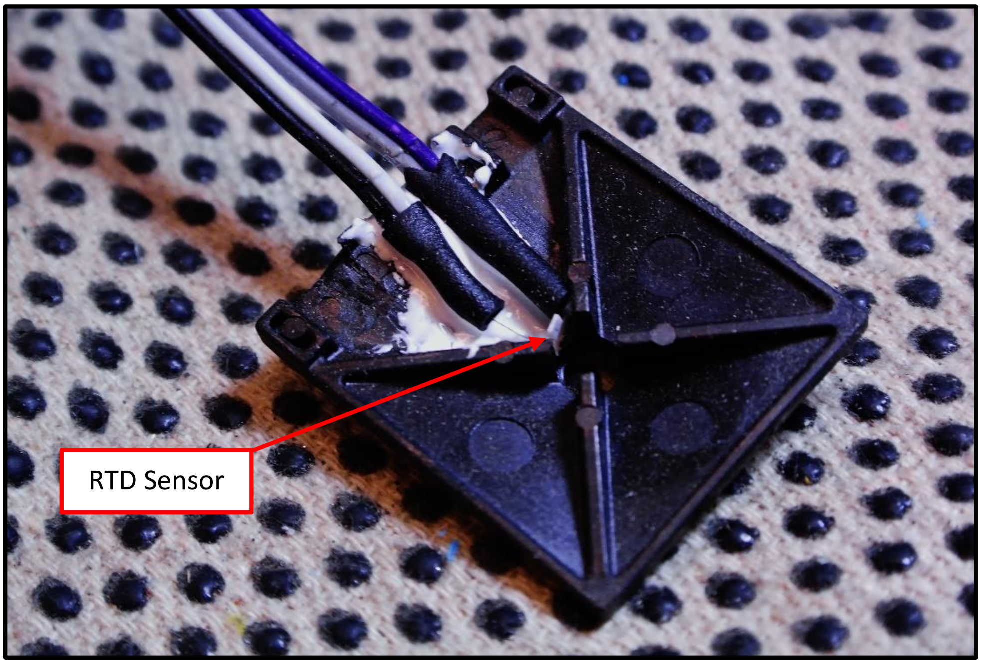


Figure 47 - RTD Implanted in Tip Shield (Shown separate from tip and without thermal grease for greater detail)

The RTD was wired to an analog to digital converter (ADC) that transmitted temperature data over a serial peripheral interface (SPI) communication link to the sentinel processor. While the RTD was slower to respond than a thermocouple, this RTD had a time constant between 1 and 2 seconds, which was well within the required design parameters for the sentinel [35]. The RTD sensor was calibrated prior to installation in the machine to ensure correct operation.

The RTD circuitry was calibrated by heating a 500 mL beaker of water on a hot plate and verifying proper temperature measurement at several set points. The temperature was stabilized with the heater control of the hot plate using thermocouple feedback to determine when a constant temperature was achieved. This feedback was provided by an Omega Type K thermocouple plugged into an Omega MDSSi8 benchtop thermocouple thermometer. For improved accuracy, a mercury thermometer was also placed in the beaker. The thermometer used was a Fisherbrand model 15-041C mercury-filled glass thermometer, manufactured by Fisher Scientific. This particular glass thermometer has a manufacturer-specified accuracy of ±0.3°C, which is superior to both the RTD and K-type thermocouple measurement systems. RTD readings on the LabVIEW interface were compared to the readings from the glass thermometer at five different temperatures. The data from each measurement device is shown in Table 4:

Table 4 - Preliminary RTD Calibration Data for Attack Testing

Set Point Temp. (C)	Glass Thermometer Temp. (C)	RTD Temp. (C)	Thermocouple Temp. (C)
0	**0.6 ± 0.1**	**0.74 ± 0.01**	**0.4 ± 0.2**
25	**25 ± 0.1**	**25.09 ± 0.12**	**24.6 ± 0.05**
50	**50 ± 0.1**	**50.17 ± 0.15**	**49.1 ± 0.05**
75	**75 ± 0.25**	**75.21 ± 0.16**	**73.1 ± 0.05**
100	**99 ± 0.25**	**99.5 ± 0.3**	**100.1 ± 0.1**

The uncertainties recorded here represent the fluctuations in the temperature readings. It is evident from these readings that the RTD data are more accurate than the thermocouple data. The reduced reading uncertainty in the readout from the thermocouple bench thermometer is suspect though, and the low fluctuation in the data is most likely due to the benchtop thermometer's on-board filtering algorithm for reducing the variance of the data output stream. Although the printing temperatures fall outside of this range, this calibration was performed in the 0-100°C range for ability to use water as the reference temperature bath since the glass thermometer requires immersion in fluid for accurate readings.

4.2.5 XYZ Position

Monitoring the printing head's XYZ position was incredibly important for defending against the easiest-to-execute attack: movement or deletion of waypoints. There exist many different ways to alter the geometry of the toolpaths sent to the printer, with some of these methods being virtually undetectable. However, if a printer is fitted with a sentinel monitor, it can check to make sure that each location was reached and the print parameters were acceptable at those locations. Most commercially-available FDM printers are equipped with stepper motors to drive each axis, and they usually run open-loop (only expensive industrial versions like the Stratasys Fortus line utilize encoder feedback). Thus, encoders had to be fitted to each axis for accurate position measurement.

The first iteration of the XYZ position sensing system implemented Avago AEAT-6012 12-bit absolute encoders. After running a significant number of tests, re-writing the driver code, and then isolating the encoders out of the system for testing, it was determined that the encoders were simply producing an unreliable signal. This variance was ultimately found to be the fault of the magnetic compass chip used in the AEAT-series encoders. The spacing of the magnet from the magnetic sensor is incredibly important for reliable data readout, but this is extremely difficult to regulate inside of the printer. It was therefore decided that switching to optical quadrature encoders and fixing the codewheel to the motor shaft was preferable. After researching encoder options, the Avago HEDS-series encoder was found as a reliable quadrature option. It also afforded the ability to be mounted intermediately on a shaft, which enables direct motor mounting for both the X and Y axes. An Avago HEDS 9140-A00 quadrature encoder was therefore chosen as the replacement for the unreliable AEAT-series encoders:

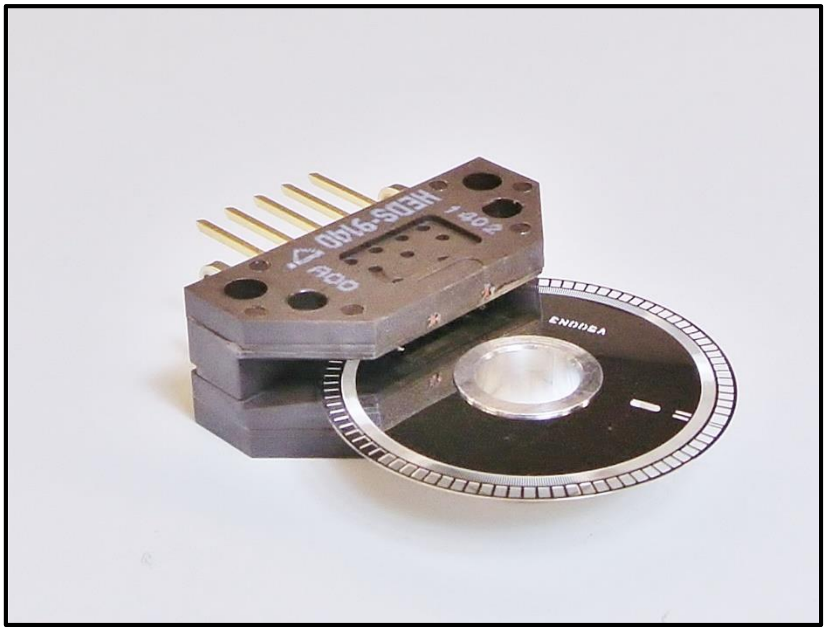


Figure 48 - New Avago HEDS Series Encoders for X, Y, and Z Axes

With intermediate shaft mounting abilities, the new encoders fit into the tight spaces not previously accessible with the AEAT-series encoders. Direct mounting to the X motor was easily achievable, but the Y motor shaft needed an extension. The motor shaft is accessible from the rear, but it is not long enough to extend from the housing. An extended motor shaft was turned on a lathe to mount the codewheel to the Y axis motor shaft:

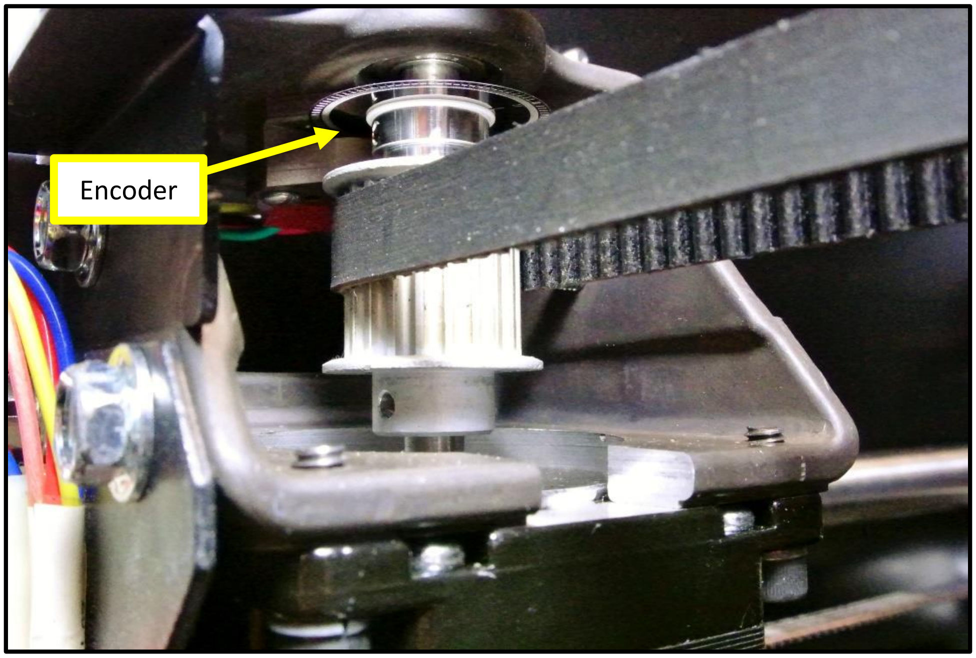


Figure 49 - X Motor Encoder Final Mounting Scheme

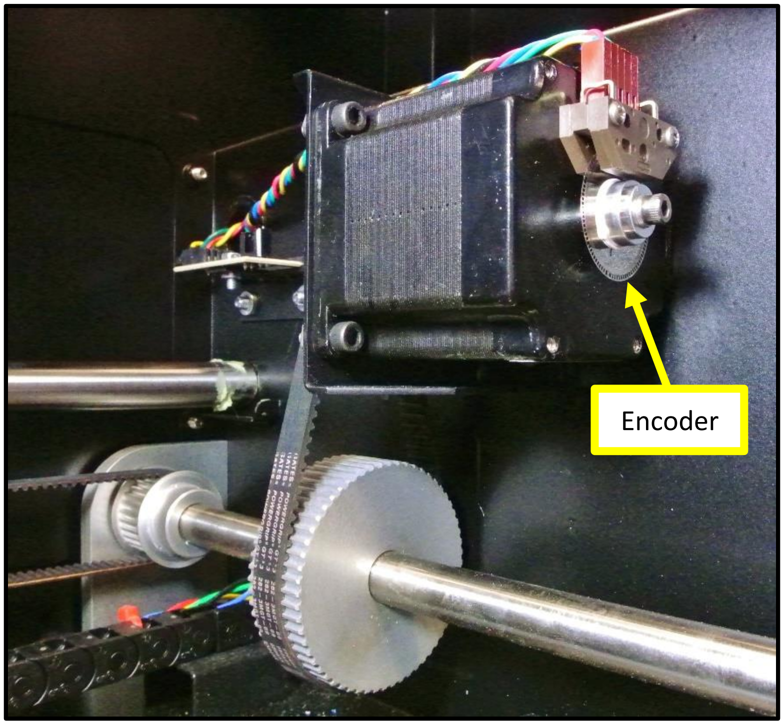


Figure 50 - Y Motor Encoder Final Mounting Scheme

Initial iterations for the Z axis positioning used the Avago AEAT-6012 encoders as well, but their failure led to the implementation of a hacked digital caliper for digital readout on the Z axis. The calipers initially worked well for accurate Z positioning, but they ultimately could not operate for long periods of time in the 77º C build environment. After several days in the heated enclosure, the caliper-based system developed a drift inaccuracy and the sampling rate dropped significantly. With the Avago HEDS-series encoders able to withstand temperatures above 100º C, it was decided that a new position measurement system should be designed for the Z axis around these optical encoders. Unlike the capacitive linear encoder signal from the calipers, the optical signals from the HEDS-series encoders are not significantly affected by elevated heat. The difficulty for the Avago HEDS encoders lies in the ability to mount the encoders to the Z-table. An idler shaft utilizing rack and pinion gearing was designed for the Z axis, and the system was CNC machined from 6061-T6 aluminum. The system is shown in the figure below:

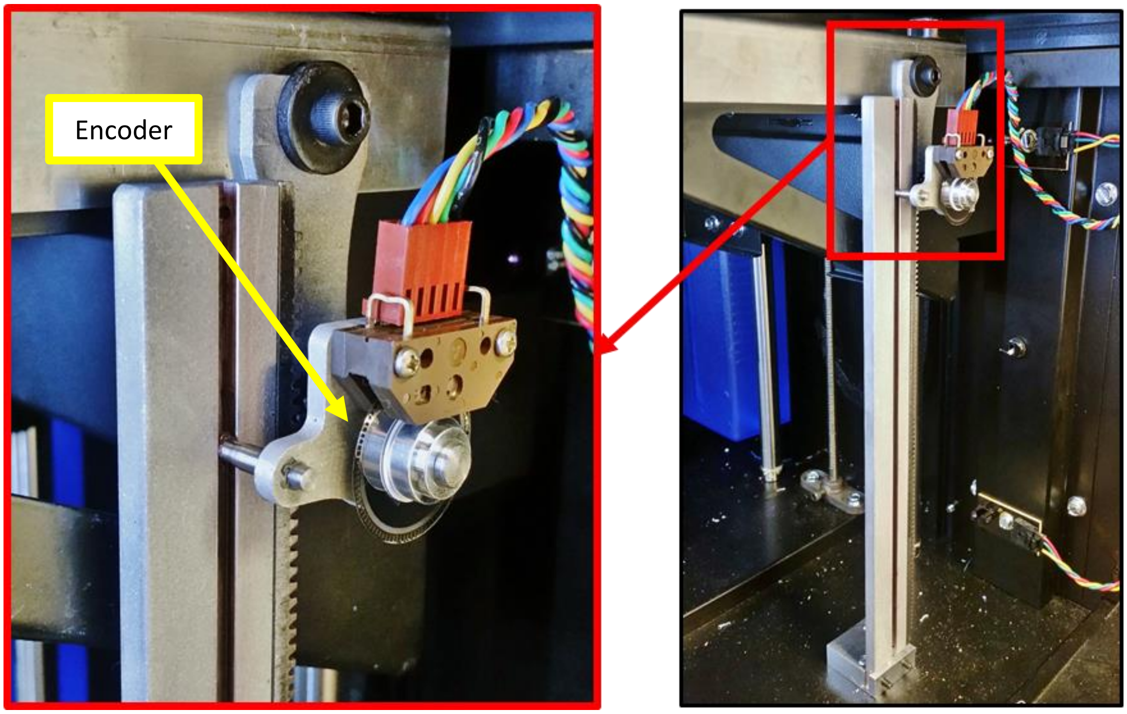


Figure 51 - HEDS Encoder Mounting to the Z Table

The new Z axis measurement system handled the 77º C temperature of the build environment quite well. The backlash in the rack and pinion drive system caused a small error in the position readout, but the overall accuracy of the encoder system after calibration was ± 4 mils, a vast improvement over that of the AEAT-series encoder system. With the new encoders mounted to the drive mechanisms, it was necessary to write a new encoder checking code. The assembly driver code was much simpler than that for the AEAT-series encoders due to the simpler transfer protocol for the HEDS-series encoders. See Appendix A for this code.

4.3 Sentinel CPU Design

The logic controller for the sentinel monitor had several important performance requirements. First, it needed a relatively fast clock frequency to enable reading of the quick quadrature encoder feedback. Ease of assembly programming was also important, since interpreted languages can waste precious processor resources. The microcontroller also needed parallel processing abilities. With the many different sensor readings, a single processor with interrupts would not have achieved the speed requirements in a deterministic way. Therefore, only multicore processors were considered.

Since feedback signals were 10, 12, and 24-bit numbers (encoders and RTD ADC output), it was also helpful to have a 32-bit processor to enable storing these numbers in one register, rather than having to split registers up and perform 2 to 4 times as many operations. The processor also needed a sufficient number of General Purpose Input/Output (GPIO) pins for sensor/controller interfacing. This was a minimum of around 20 pins.

One important function of the logic controller was to interpret the CMB files to retrieve the build parameters. The fixed-point method for simplifying calculations removed any requirements for the processor to have floating point hardware (which is less common and found in more expensive digital signal processing (DSP) microcontrollers). Reading the CMB files directly meant that there must have been some way to transfer the CMB file to the logic controller, namely some sort of flash memory interface. The ability to read a stored file from an SD card is desirable for this reason.

The Parallax Propeller microcontroller was chosen for these operations. The Propeller setup was overclocked to 100 MHz, making it sufficiently fast for reading samples from each of the sensors. The Propeller chip itself is composed of eight parallel 32-bit processors, allowing simultaneous reading and calculation of the individual sensor data. The Propeller also boasts 32 GPIO pins, which supported the sentinel’s hardware interface. It also allowed SD card reading and writing, enabling the transfer of CMB files from a host computer to the sentinel. The SD reading method enabled the Propeller to read waypoints directly from external flash memory, providing parallel position-checking and CMB-reading operations on large files. Most importantly, the Propeller chip is relatively inexpensive; it can be purchased in bulk for around $4.

The Propeller supports a variety of different languages, but the sentinel code developed through this work was an amalgamation of interpreted Spin code and Propeller assembly code. Spin is a proprietary language developed for the Propeller chip, and while it works well with the Propeller chip, assembly code is executed much more quickly than Spin, with the caveat of being much more difficult to write. Assembly code was used in the sentinel code where speed requirements could not be met with interpreted code.

4.4 Assembly Encoder Driver and Parameter Checking

The HEDS-series quadrature encoders employ a simple interface to the sentinel CPU. There are two square wave pulse input channels per encoder (one pulse per channel per unit movement). Three encoder drivers were written: one for the XY encoders, one for the Z encoder, and one for the extruder encoder. These codes ran on parallel processors to maximize the read speed of the Z and extruder encoder driver. The XY encoder driver started by reading the state of the X encoder’s output pins, then checking for a change from the previous output state. If there was no change, it moved to the Y encoder. It then watched for each of the four possible quadrature output cases (there were 2 output bits, and therefore 2^2 possible output cases). The quadrature output signals from the encoder are 90º out of phase from each other, so one of the sensors has to trigger first. The assembly code driver inferred rotation direction from these signals by testing which sensor output a high pulse first. It also calculated the angular velocity of the encoder by reading the pulse width of the encoder output channels.

4.4.1 Use of Fixed-Point Notation for Waypoints

One major feature of the encoder driver was the implementation of fixed point notation in the code. It was determined that mil-accuracy (±0.001 in) was more than sufficient for the checking algorithm, so a scaling factor of 1000 was chosen and all floating point values were replaced by fixed-point values. The SD card reader code was re-written to convert and accept this new fixed-point format. Multiplication and division functions were also written into the assembly encoder codes to enable scaling the axial positions to the fixed-point format. The motivation for the fixed-point conversion was this: fixed-point decimals do not require the use of an extra processor for the floating-point driver. In fact, assembly methods can perform operations on the fixed-point values in the code, since they are stored as signed integers. This allowed for the removal of the floating-point driver cog (processor), the FloatMath object, and the Spin method running the floating-point calculations. Deleting all of these methods freed a large amount of valuable processing power and increased the speed of the operations. Converting the positions and build file waypoints to scaled, fixed-point format allowed the parameter checking algorithm to move into assembly code, increasing the speed dramatically from a Spin version of the same code.

4.4.2 Assembly Checking Method Structure

Parameter checking speeds increased dramatically upon conversion of the parameter checking method to assembly. To increase the overall speed of the parameter checking algorithm, waypoint reading was also revised. The old version of the checking algorithm read the parameter set for one point, waited until that point's parameters were achieved, and then retrieved the next parameter set from the SD card. These SD read operations took a fair amount of time, and reading sequentially slowed down the algorithm considerably. A buffer was therefore implemented into the code to help speed up parameter checking. When the SD method was called, the code retrieved the next 50 points instead of just the next one point. Data was loaded into global memory (Hub RAM), then the code started an assembly checking cog, loading the parameters for each point onto the Cog RAM (local memory for that processor) where they remained for the duration of that assembly cog's run. Meanwhile, the next 50 points were retrieved from the SD card and stored in Hub RAM. A second assembly checking cog was then started with the next 50 coordinates, but it waited for a flag variable to be set by the first cog. When the 50th data point was checked off the list, the first assembly checking cog set the flag variable and halted execution. The second assembly cog read the flag and began checking points 51 through 100. This loop continued until the end code in the build file was reached.

4.4.3 Adding Tolerances to the Waypoints

Each waypoint in the CMB file was sent to the uPrint's motherboard. However, the motherboard rounded these values to the nearest thousandth of an inch (1 mil), providing a source of rounding error, E_R. There was also an error associated with the random deviation from the encoders, E_{dev},which is $\pm$ 3 mils. These errors combined according to the total variance formula to yield the error from the curve fit for the encoder calibrations. The errors exhibited in the X and Y calibrations, E_{cal}, were rounded up to $\pm$ 4 mils. To allow for slight movement errors, the tolerance band for accepting a given position was elevated above this 4 mil minimum to a liberal 10 mils. This allowed for early acceptance of points within a 20 mil-wide square of the expected waypoint, giving the next point time for acceptance. The minimum feature size in the crack specimen was approximately 30 mils, so this acceptance algorithm still detected these features. In the sparse in-fill, filaments spaced 24 mils apart will still bond to one another, so this detection method was able to easily detect if adjacent filaments were bonded together or had an air gap between them.

4.5 Sentinel CPU Wiring and Sensor Interface

A prototype sentinel monitor was built for testing in the 3D printer use case. This prototype was built first on a breadboard, then soldered to perf board as a more permanent solution. The wiring diagram is shown below:

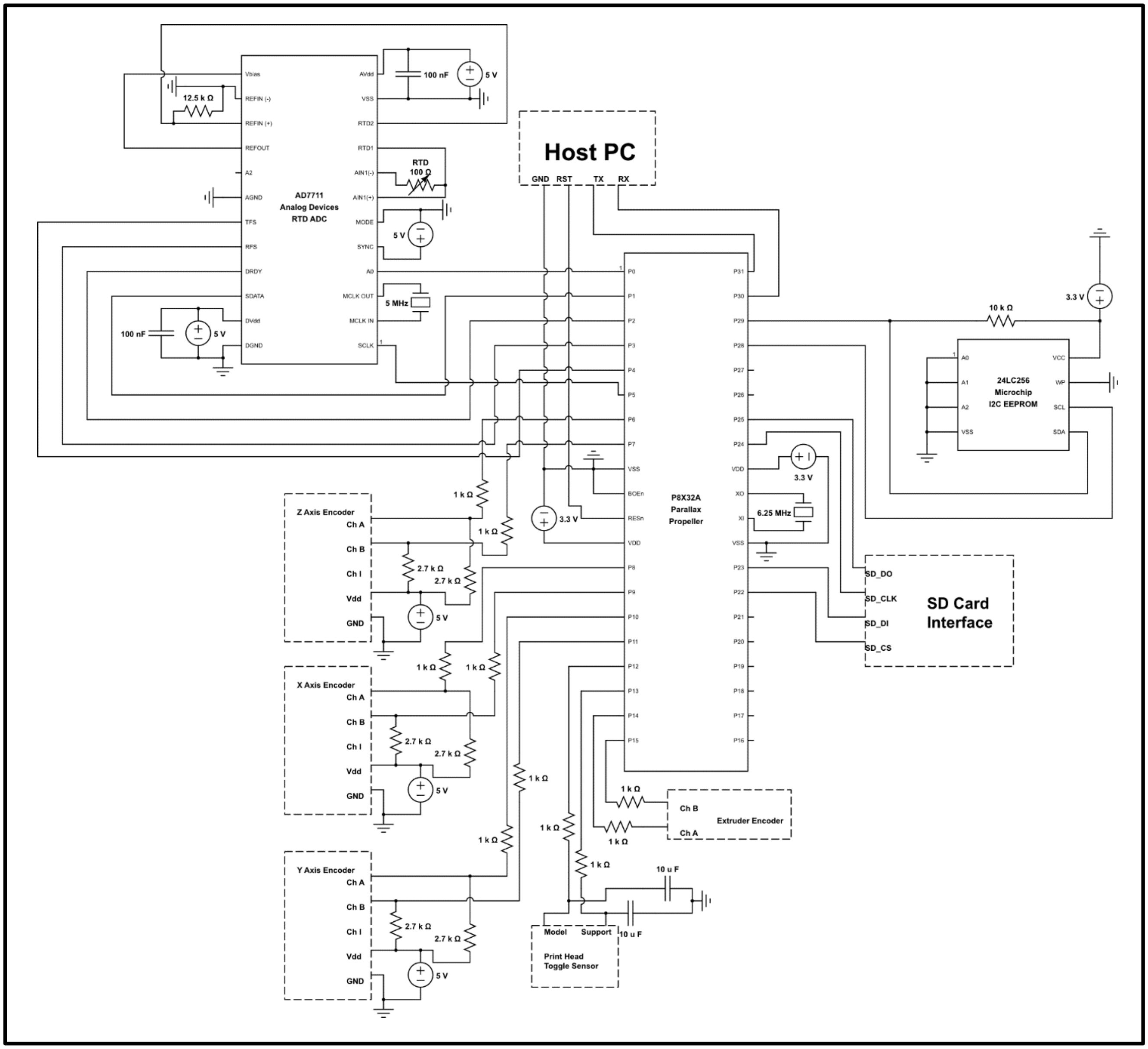


Figure 52 - Wiring Diagram of Prototype Sentinel Monitor

The AD7711 RTD ADC uses an SPI interface, but it requires a few extra pins to set its internal control registers. These control registers control the output mode of the ADC and change certain behaviors such as filtering and sampling frequency. Pins 30 and 31 on the Propeller were reserved as RX and TX for serial communication with the host PC. This is mainly helpful for debugging code, since RAM-addressed registers can be sent to the host PC over this connection.

The perf-board version of the sentinel CPU is shown in the figure below:

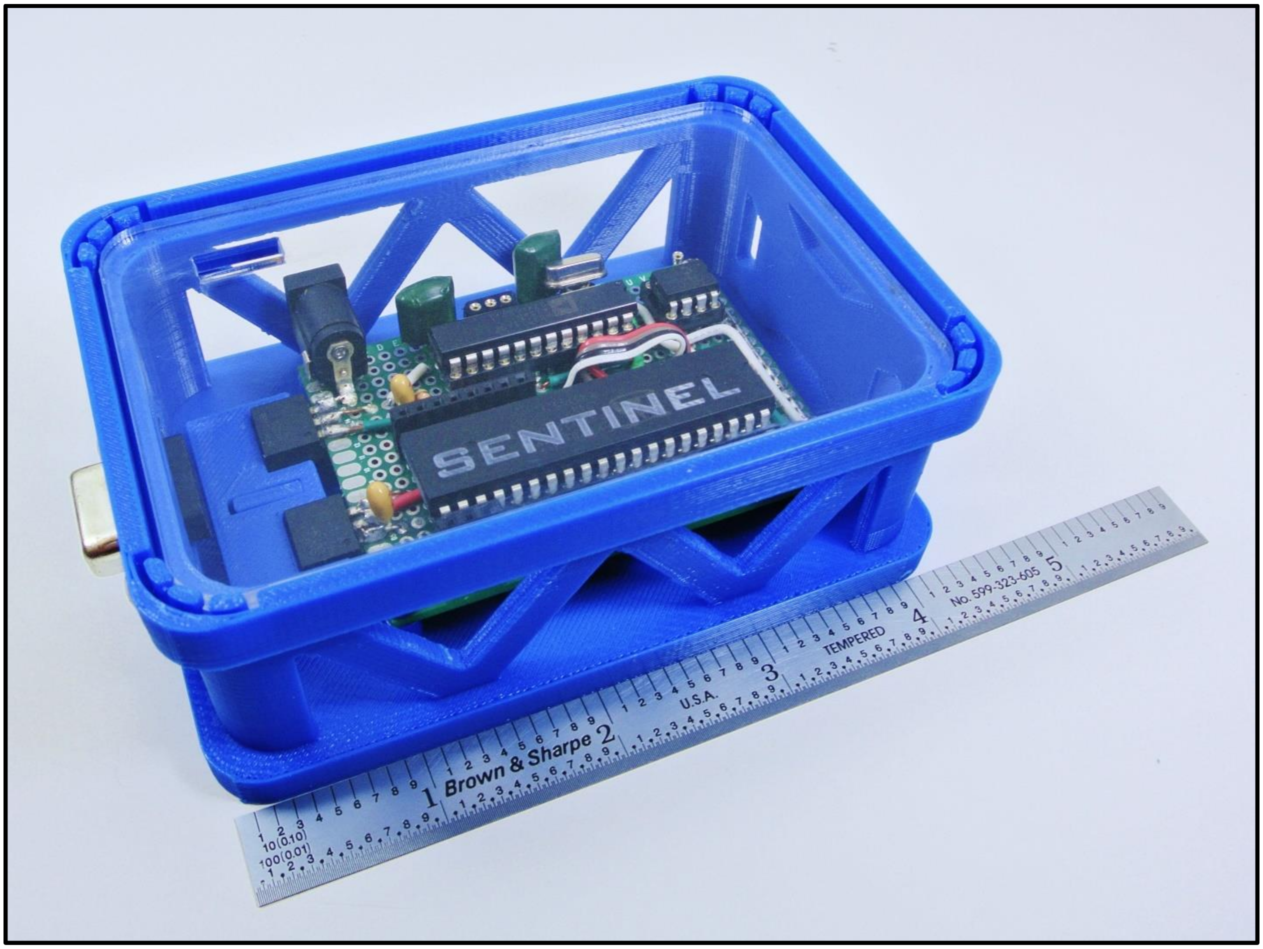


Figure 53 - Perf-Board Version of Sentinel CPU

While the sentinel was designed as a self-contained unit, it is noteworthy that this board could be integrated into the printer’s hardware as long as proper isolation procedures are taken to ensure the system security of the sentinel device in the event of a machine hack.

4.6 The Importance of Cost

The individual technologies at work inside of this sentinel device have existed for decades, but the widespread implementation of such a sentinel device has recently become much more feasible due to a drastic reduction in cost of implementing these technologies. It was determined during the design stage that the entire sentinel device should not exceed $250 in parts to maintain feasibility of the use of a sentinel monitoring device as a security measure. The costs of all the individual parts used are listed in the following table:

Table 5 - Parts List for 3D Printer Sentinel

Quantity	Description	Unit Price	Line Total
3	Avago 9140-A00 Encoder	$ 29.53	$ 88.59
3	Avago 5140-A06 Codewheel	19.95	59.85
1	AD7711 RTD ADC	38.77	38.77
1	Propeller Chip	7.99	7.99
1	I2C EEPROM	1.50	1.50
2	Linear Voltage Regulators	1.50	3.00
1	RTD Sensor	22.00	22.00
		Subtotal	$ 221.70

There are of course extra costs, such as hookup wire and circuit boards, but large scale sentinel production would substantially reduce the overall manufacturing cost. The marginal price increase of even $250 to this $25,000 machine justifies its inclusion for secure system operation.

This design process encourages the application of a system monitor to 3D printing systems to ensure proper printing behavior. The low cost of this sentinel monitor also allows for the possibility for this system to be used in a quality assurance manner. There are currently no methods for prediction of fracture behavior for 3D printed parts based on monitoring the printing parameters.

4.7 A Robust Design

Even though low cost was a major design parameter, only trustworthy, industrial-grade IC chips and sensors were used in the final version of the sentinel device. The flaws encountered in the initial absolute encoder system challenged the robust nature of the design and therefore needed to be addressed. The new HEDS-series encoders contribute to a robust mechatronic design for the positioning system.

The thermocouple-based temperature sensor employed by the uPrint also showed signs of irregularity. The RTD used in the sentinel was chosen for its improved accuracy and reliability over this thermocouple measurement system. The sources of variance were actually somewhat surprising for the thermocouple system; the Type K thermocouples used have a specified accuracy of either ±1.1º C or ±2.2º C, while the Analog Devices thermocouple amplifier IC chip specified another 4º C of inaccuracy [36]. The temperature experiments yielded significant changes in strength even at 10º C change in tip temperature, proving that the thermocouple measurement system is non-ideal for this environment. The RTD temperature measurement device, while nearly 4 times as expensive to implement as a simple thermocouple measurement circuit, provided much more accurate temperature feedback with less random noise in the output. The specified accuracy for the RTD is ±0.15º C, while the accuracy for the RTD ADC chip is ±0.45º C [37]. The disadvantage of using RTDs is the slower response time, but this particular RTD has a time constant of 1.5 seconds, which is acceptable considering the overall thermal mass of the heater assembly. This hardware alteration was a trade-off: better accuracy and a steadier output signal at the expense of a speedy response.

To maintain this high level of integrity in the sentinel device, each section of the sentinel code was tested both individually and in combination with the entire sentinel system. Drivers for any malfunctioning subsystems were isolated from the sentinel code for debugging. This expedited debugging processes due to the isolation of variability.

The dynamic properties of the printer also taxed the software limitations of the sentinel, requiring each method's optimization for speed. The axial speeds of the gantry head caused positions to change a few mils per millisecond, meaning that position updating and checking had to occur even faster. Of the eight individual methods running on the sentinel device, five of those methods were written in assembly. The faster execution time afforded by assembly code necessitated its use, even though debugging the assembly code took much more time and effort than debugging Spin code would have taken. Designing such a complicated system requires sacrificing some performance characteristics to meet the performance requirements in other areas. Thus, ease of programming and debugging was sacrificed for faster operating speed.

Chapter 5 – Sentinel Testing

5.1 Building a Foundation – Initial Testing

Before running initial tests on such a complicated system, it was important to test each subsystem individually, and the first subsystem chosen for testing was the XYZ positioning system. Bench tests were run to display the current encoder position, starting with just one axis and then expanded to include every axis. I/O signals were monitored on an oscilloscope, and position data was read into the Parallax Serial Terminal (PST), a simple serial interface between the Propeller chip and the host PC. When all three encoders correctly tracked idler shafts on the bench, they were then installed into the machine and interfaced with their respective drive systems. Tests were then run with the encoders installed in the machine. Movement commands were sent to the uPrint via a diagnostics port on the rear. A USB-to-RS232 converter cable was used to bridge the host PC with the diagnostics port on the printer, allowing direct communication with the uPrint’s motherboard. This allowed direct control of the X, Y, and Z stepper motors with the ability to move them to a set coordinate (in inches zero-referenced from the machine’s own zero location). Duplex communication with the uPrint was achieved through a serial terminal (Tera Term) that was calibrated to the proper serial protocol for the machine.

5.1.1 Calibrating the XYZ Axes

While the assembly driver retrieved the XYZ positions, it returned them in units of encoder counts. The XYZ positions had to be calibrated against machine coordinates and converted to inches. Machine coordinates were obtained using Tera Term (a freeware serial terminal interface program) and the serial diagnostics port. The head was moved to a particular XYZ position using the “Move X”, “Move Y”, and “Absolute Z” commands in the terminal. With the relationship between these machine coordinates and the encoder counts, it was then time to translate the encoder position feedback into the scaled, fixed-point format. Linear fits were taken from the data in Excel and applied to assembly-language formulas on the Propeller chip. These fits are displayed in the plots below:

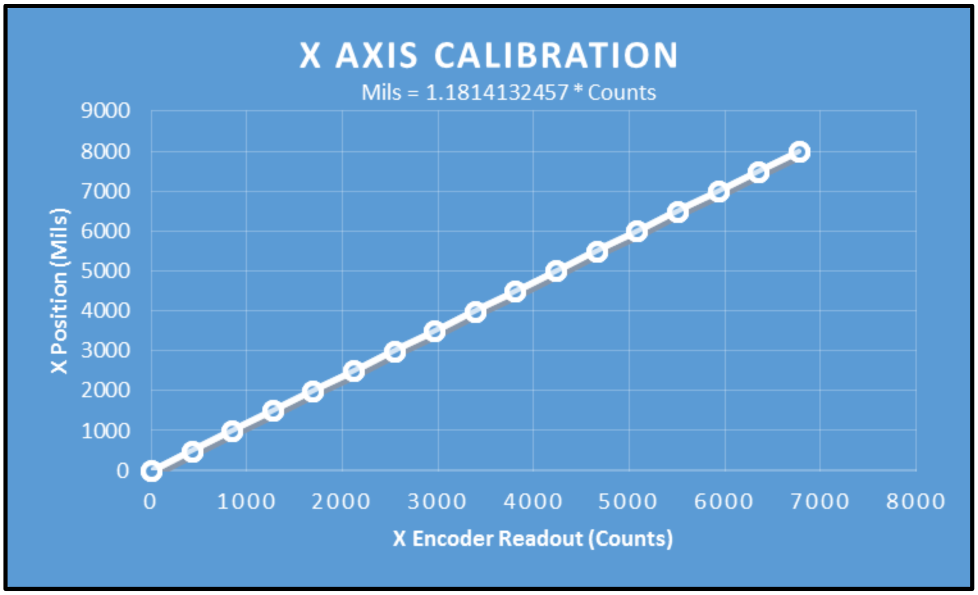


Figure 54 - X Axis HEDS Encoder Calibration with Line of Best Fit

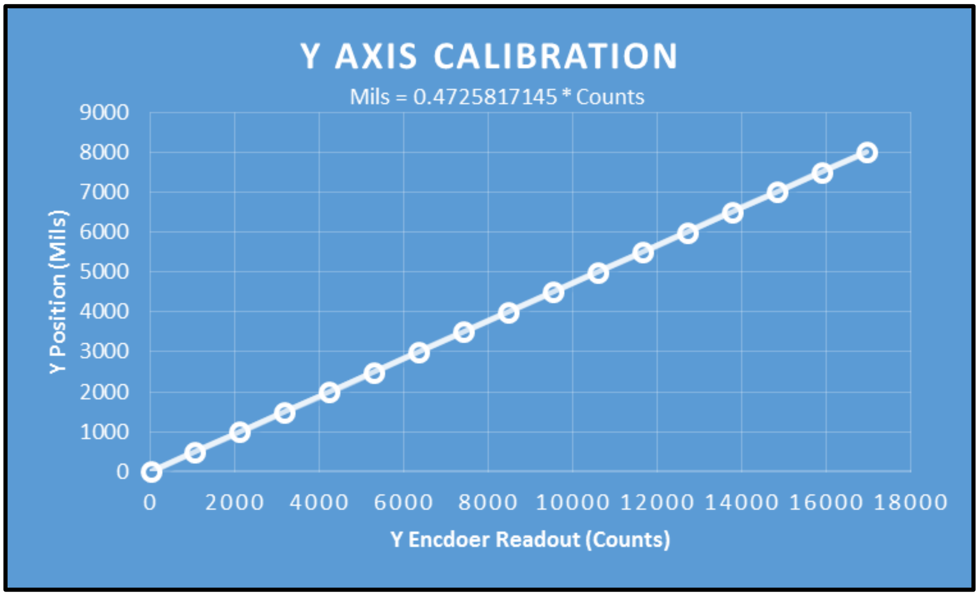


Figure 55 - Y Axis HEDS Encoder Calibration with Line of Best Fit

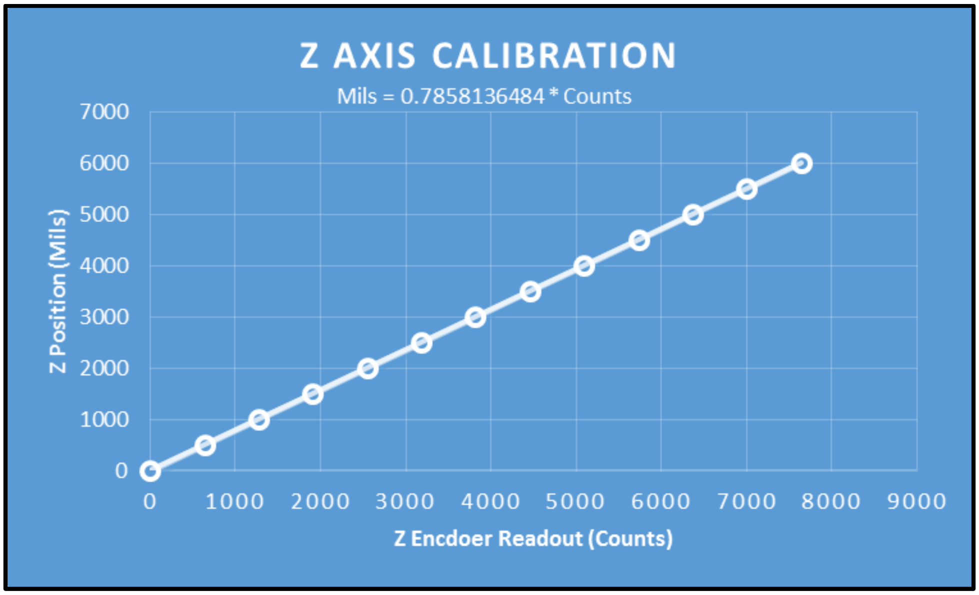


Figure 56 - Z Axis HEDS Encoder Calibration with Line of Best Fit

5.1.2 Adding Extrusion and Material Usage Detection

The material and extrusion detection methods were tested upon successful calibration of the XYZ positioning system. A low-pass filter circuit was added to the material detection circuit to smooth out high frequency noise in the signal. Initial extrusion detection methods worked properly, detecting proper extruder status. Testing was therefore expanded to include sentinel monitoring of a simple print process.

5.1.3 Simple Square

A simple test part was printed with this newly-calibrated location-tracking system. A build file was created to include seven waypoints to draw the perimeter of a square (some waypoints are used to jog or turn the extruder on and off). XYZ position data was logged with the encoders, and a graph of the recorded X-Y position data is shown in Figure 57:

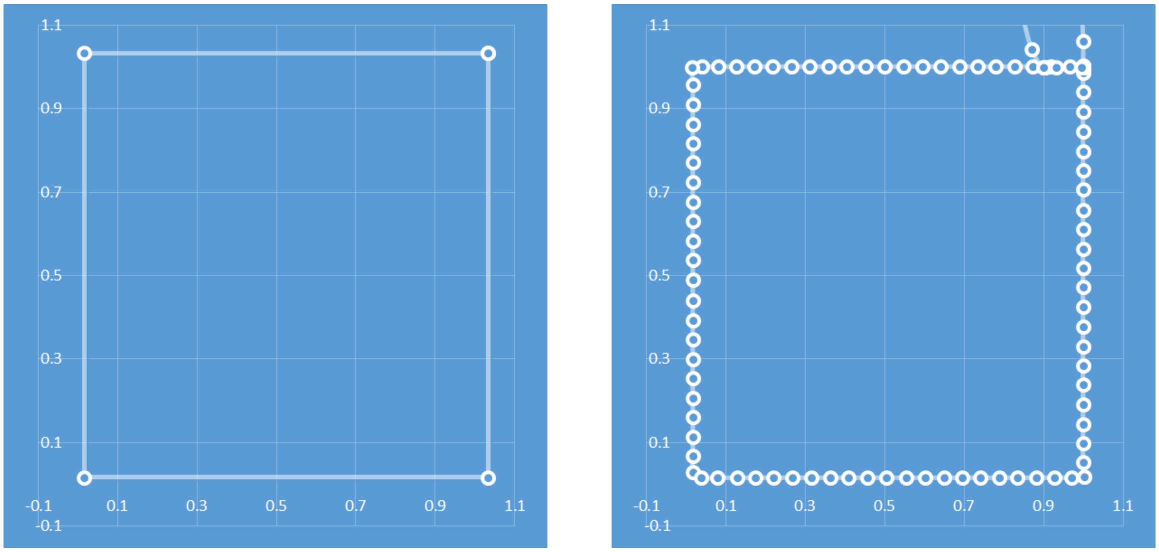


Figure 57 - First Practice Print – Expected Outcome (Left) and Recorded Outcome (Right)

As shown in Figure 57, the position data was on-point, falling in the linear interpolations between the waypoints. The extraneous points at the top were jog moves that the sentinel was expecting at that point in the process. Since this was a single layer print, the Z position stayed the same the whole time at a value of 0.013”. This test proved proper operation of the XYZ encoder system so the test was repeated with the extruder and material feedback. With these methods included, the sentinel was able to verify each point in the print file and detect the printing faults/deviations implanted in the emulated cyber attack during the printing process.

5.1.4 12-Layer Print Test

The square file above was then modified to include a support raft along with an extra layer of part material. A support raft is a sparsely-filled support structure printed between the build plate and the bottom of the model. Since a raft is a filled pattern, the extruder rasters very quickly on the interior of the part. The perimeter toolpaths (without internal fill paths) of this part are shown in the figure below:

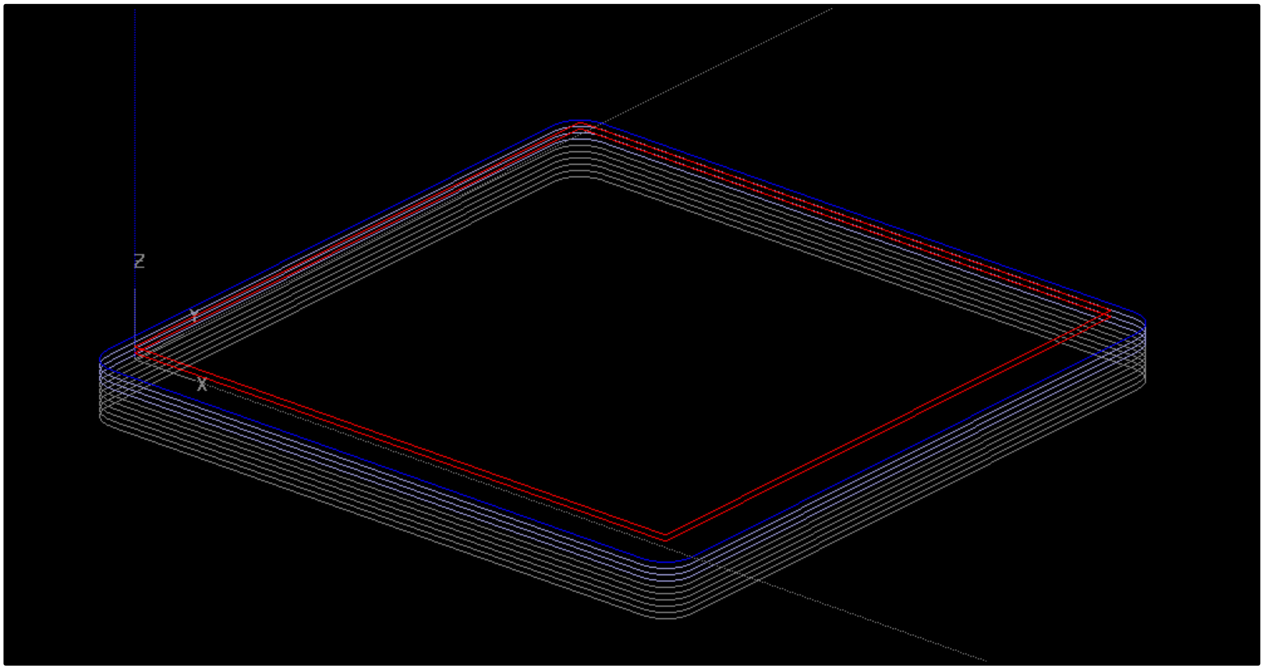


Figure 58 - 12-Layer Print (Shown in the Insight Pre-View Window) - The internal fills in the raft are not visible in this view

The internal raster patterns are some of the fastest movements in a 3D printer's motion profile and they therefore set the maximum required sampling rate for the encoders. With the assembly-driven HEDS-series encoders, all XYZ positions updated properly, and testing then shifted towards the material and extrusion detection. With assembly checking of the current print material, even the fast movements of the internal fill pattern were validated by the sentinel. They were also spaced far enough apart to produce a clean on/off signal from the extrusion detection. It was therefore time to move on to a full-scale sentinel test: the ASTM D638 Type II test samples described in Chapter 3.

5.2 ASTM Sample Testing with Sentinel Detection

With sentinel detection working for the multi-layer print job, full-scale detection was tested on the ASTM samples used during the attack testing cases. The control specimen was tested first to ensure acceptance of the non-attacked control specimen. While the material and XYZ location detection algorithms worked well for this case, it was determined that real-time extrusion detection algorithm worked improperly for the full-scale case. The higher point density of the ASTM part resulted in increased noise in the feedback signal from the extruder encoder. It was discovered that the encoder system is driven by a PMD motion processor, which applied the encoder feedback to PID control algorithms to provide smooth extrusion control despite the gantry head’s acceleration and jerk. A proper extrusion prediction method would require emulation or utilization of this motion processor chip, since it took the constant extrusion data and calculated the required variation of the extrusion rate. For this reason development of a real-time monitor for the extruder was abandoned. The high point density and tight concentration of s-curves in the interior fill of the ASTM sample caused the PMD controller to rapidly ramp the extrusion rate up and down, resulting in sporadic changes in the extruder’s rotational speed. A new, much simpler methodology was therefore applied to monitor extruder feedback via the sentinel. The sentinel was simply provided with the total distance that the extruder should travel during a part. Assuming no slip between the plastic filament and the traction wheel on the extruder, this distance is proportional to the total volume of plastic extruded by the printer.

5.2.1 Control Specimens

Two non-attacked ASTM control specimens were printed while the sentinel logged the usage of model material in the aforementioned manner to calibrate the material usage. Detecting the presence of cyber attacks during printing therefore required the sentinel to compare the material usage of the build process to that of the validation case. From the control sample, validated material usage falls in the range of 17,438,750 – 17,439,250 counts. A count is defined as a change in the quadrature output signal, thus increasing or decreasing the encoder position. With the checking algorithms working and the unhacked samples approved by the sentinel, testing of the compromised ASTM specimens started.

5.2.2 Switching Model Material for Support Material

Execution of the first attack resulted in the sentinel's rejection of the print job. While the sentinel received the validated file, the printer received a file with one small portion of the interior fill switched to support material. The sentinel failed to accept any points of the interior fill pattern with incorrect material usage. The printing defect resulting from this cyber attack was therefore successfully detected.

5.2.3 Reducing the Road Width

The second attack was detected using the extrusion volume method outlined above. The file sent to printer contained an altered internal fill in a small portion of the gauge section in which the road width was altered to half of that of the control specimen. All of the XYZ positions were met during the printing process and the sentinel accepted each location. However, at the end of the job, the overall material usage deviated from that of the control case with a significant decrease in volumetric extrusion. The altered road width introduced into the internal fill pattern through this cyber attack was successfully detected due to this variance in overall extrusion.

5.2.4 Inserting a Notch Into the Part

This attack test turned off the extruder at a specified location in the internal fill pattern, creating a small notch inside the model. When this altered file was sent to the printer, the sentinel detected proper material usage and achievement of XYZ coordinates. The extrusion volume parameter, however, was skewed slightly. The build process for this attack recorded an extruder travel of approximately 17,428,000 counts. This demonstrates a reduction of approximately 11,000 counts from the unhacked sample, with the extruder travel falling outside the acceptable range. This print job was therefore found invalid by the sentinel monitor and this cyber attack was detected.

5.2.5 Inserting a Seam into the Part

The fourth attack proved undetectable without precise detection of the current extruder state. To detect an additional part seam, the sentinel must be able to tell if the extruder halts during a certain toolpath. Since the overall material usage is not affected by an additional part seam, the extrusion volume measurement does not indicate extruder stoppage. Detection of this attack thus requires deeper knowledge of the control system governing the extruder behavior as well as more complex sentinel code. The sentinel used in these experiments was designed for simplicity, high sampling rate, and low cost. Inclusion of reliable real-time extrusion checking would drive the cost of the sentinel device up given the more complicated nature of system that the printer utilizes for extruder operation. Thus, the development of a more complex sentinel method for extruder checking is left for future research.

5.2.6 Reducing the Extruder Temperature by 20º C

Detection of the fifth and final attack requires installation of the temperature detection equipment and interfacing the temperature-checking software with the current sentinel code. The code required small changes for pin declarations and overall simplification of the operation, but completion of these changes provided the sentinel monitor with accurate tip temperature feedback. The temperature print logs and stress/strain plots from the ASTM testing were examined to determine appropriate acceptance ranges for the sentinel definition. The shortened samples used in the prior testing were also used for the detection experiments. Print temperatures were found to vary by approximately 5º C throughout the build process, requiring a range to be specified for the sentinel's temperature acceptance criteria. With the control print temperature of 210º C, the stress and strain plots exhibit a reduction in strength and elongation at 200º C, while the apparent strength and elongation at temperatures up to 230º C showed significant increase. The 230º C sample showed significantly better mixing than the control sample, and the flexural delamination (or interlayer bond) strength increased drastically. The acceptance range for the model temperature was therefore set between 200º C and 230º C. For the control case, the model tip temperature averaged 213º C in the grip section and 208º C in the gauge section of the specimen, well within the acceptable range. However, when the attack case was introduced and the model tip temperature was adjusted to 190º C, the sentinel failed to accept the first point that read under 200º C, rejecting the attacked part. The sentinel therefore successfully detected a potential cyber attack on the extruder's temperature setpoint.

5.3 Sentinel Testing Results

These tests proved that a sentinel monitoring device can properly detect the existence of printing faults in a 3D printing process. While the fourth attack went undetected, more complicated sentinel hardware and software could solve this issue. Better extruder checking methodology and understanding of the uPrint's hardware and parameter manipulation would be required for more robust real-time extruder feedback. The XYZ position, material, and temperature checking systems, however, properly detected their respective attacks. The fast sampling rate achieved through the use of assembly code combined with reliable sensors afforded the ability to check highly point-dense files for printing faults.

5.4 Limitations of this Study

The prototype sentinel monitor described in this thesis served as an initial proof of concept for the application of System-Aware Cyber Security solutions to additive manufacturing techniques. However, this study focused only on one 3D printing technology: Fused Deposition Modeling (FDM), also referred to as Fused Filament Fabrication (FFF). FDM was chosen for this application due to its popularity and relatively low cost. FDM printers are used predominantly for home, office, and engineering design, making them arguably the most widelyused 3D printers, producing models with some of the strongest mechanical properties of any polymer-based additive manufacturing process [9]. The principles applied to the uPrint sentinel monitor have laid a foundation for sentinels protecting other 3D printer technologies. In its current state, the sentinel monitoring system needs to be customized to each printer, requiring extensive setup time and wide variances in cost from machine to machine.

Only the five most obvious attacks were considered for protection. The sentinel hardware and software was designed to monitor only the affected parameters, leaving some attack vulnerabilities in the prototype system. A fully-realized sentinel monitor would monitor all print data, not just the relevant parameters monitored by the prototype system. One way to improve these traits would be to add more precise sensor calibration or more expensive sensors and transducers. The lab equipment used for RTD calibration was unable to replicate the 230º C conditions seen in the extruder tip (boiling water was used for calibration, which could not exceed 100º C). Thus, points for the calibration curve were gradually increased to the maximum temperature that the calibration setup could support, and the RTD ADC calibration registers were set there. A hotter and more stable heat source would yield more accurate values for the RTD ADC calibration registers throughout the entire experimental temperature range.

Chapter 6 – Conclusions and Future Work

6.1 Security Concerns for Advanced Manufacturing

The work summarized here is a case study for the System-Aware Cyber Security solution proposed by Jones and Horowitz specifically applied to an additive manufacturing system [15]. The current shift toward the use of additive manufacturing technologies from proof-of-concept prototypes towards the fabrication of critical components necessitates the security and quality assurance of 3D printing processes. In use cases such as 3D printed airplane wing spars, model failure must be avoided at all costs. Malicious attacks on machines creating structural components such as these could result in major losses of life.

A computer-based sentinel device can react to printer operations much more quickly and precisely than any human-based monitoring system. The mechatronic nature of 3D printing technologies requires new security procedures since physical outputs are not monitored through traditional perimeter and network security methods. The case study previously presented by Jones and Horowitz involved the application of a sentinel monitor to a UAV control system, demonstrating the basic ability of the sentinel to search data streams for illogical data outputs that resulted from logical data inputs [15]. A 3D printer, however, takes data inputs and transforms them into physical outputs, making small process deviations difficult to detect. These physical outputs must therefore be measured with properly-calibrated sensors implanted in the printer. Thus, our definition of a sentinel monitor expands to include not only a logic controller that processes input and output data streams, but also any related peripheral devices, including sensors and instrumentation IC chips required to interpret the sensor data.

These findings place the security responsibility on either printer manufacturers or private sentinel manufacturers. A typical consumer would be unable to develop the required sentinel hardware and software without detailed knowledge of the internal working of the machine and advanced manufacturing capabilities. For example, each encoder mount for this sentinel application was fabricated in a machine shop on precision equipment to ensure the proper alignment of the encoder codewheels and sensors. Temperature sensors were carefully secured with aluminum-oxide epoxy to withstand the high temperatures experienced by the extruder tip. The sentinel protection algorithms also required in-depth knowledge of the printer's architecture. For the printer manufacturer, however, inclusion of a secure monitoring system is easily achievable. Original Equipment Manufacturer (OEM) equipment could be built with the sentinel as a design consideration, easing the burden of custom instrumentation.

6.2 Future Work

This preliminary research effort was successful in demonstrating the risk of cyber attacks to additive manufacturing processes and the testing of a prototype monitoring system (sentinel_ to detect such attacks. Further work is needed to explore the following related options, the study which were beyond the scope of this project:

6.2.1 Extension to Metal Printing

With the increasing interest in 3D printing components for assemblies, the security risks associated these mechatronic systems must be addressed. The next logical step when expanding the System-Aware Cyber Security solutions proposed in this paper would be to apply it to the design of a secure sentinel monitor for industry standard metal 3D printers, such as those made by Arcam or EOS. The security vulnerabilities in metal 3D printing technologies are extremely relevant to this study because they afford a cyber attacker the ability to induce catastrophic failures in critical metal components. Modification of the uPrint sentinel designed through this research for use in metal 3D printers would require a new array of sensors, sensor drivers, and sensor/processor interfaces. However, the overarching concepts explored in this study are directly applicable to metal printing techniques. Many analogous build parameters exist between FDM and metal technologies (tip temperature versus laser intensity, beam positioning versus extruder nozzle positioning, etc.) and the monitoring process developed for the sentinel could be adapted for a metal printing process.

6.2.2 Extrusion Rate for FDM Sentinel

The next step in the development of a better sentinel design would involve improvement to the current extrusion rate measurement and prediction system. The prototype sentinel was unable to properly predict the printer's extrusion operation because the gains used on the on-board motion controller were not precisely known. With deeper knowledge of the printer's calculation of the extrusion rate based on the specified road width and the dynamic behavior of the extruder, the sentinel would have been able to properly predict and detect all changes to the extrusion rate during printer operation and therefore be able to successfully detect the seam attack Further research is therefore needed for the inclusion or emulation of more complicated system architectures on the sentinel device itself.

6.2.3 Diverse Redundancy

Diverse redundancy is the parallel operation of two boards of different architecture. Dynamically hopping between outputs from each of these boards protects the security of the system by making the entire system more difficult to enter (attackers must learn the details of each individual system or gain influence in multiple suppliers before they can attack the system). Printer security could be further increased through the implementation of a diversely redundant sentinel. Since sentinel hardware is inexpensive, cost increase due to implementation of diverse redundancy would be marginal. Further research focused on the application of diverse redundancy techniques to a 3D printer sentinel would more effectively shield sentinel operations from cyber attackers.

6.2.4 Fault Isolation

Another important area for further research is fault isolation. Fault isolation is defined as the ability of the machine to precisely detect a printing fault's origin, effects, and suggested remedial actions. When this prototype uPrint sentinel detected a printing fault, the sentinel ceased its operation and recorded the first point that exhibited unacceptable build parameters. Applying fault isolation techniques might expand this behavior to alert the operator of the subsystem in which the fault originated. However, some build parameters might carry more weight than others, as illustrated through the tensile test results from the attack specimens. A point system could help quantify the severity of the attack and its affect to the fracture strength (points assigned for the amount of deviation from the expected toolpaths, temperature differentials, etc.). More research is also needed to determine appropriate remedial actions. For instance, notification of a printing fault should signify certain repairs to the machine or network security, with attack severity governing these remedial actions. The desire to reduce machine downtimes drives the need for fast responses to malicious attacks. In large-scale manufacturing facilities, printer inactivity due to industrial sabotage could be catastrophic. The inclusion of suggestive corrective actions after an attack has been detected would therefore be a desirable feature in a next-generation sentinel monitoring system.

References

[1] Albright, D., Brannan, P., & Walrond, C. (2010). Did Stuxnet Take Out 1,000 Centrifuges at the Natanz Enrichment Plant?. *Institute for Science and International Security.*

[2] (2014). 3D printing company cited by OSHA after explosion, facing $64,400 in penalties [Web log post]. *Retrieved from: http://www.3ders.org/articles/20140522-3d-printing-company-cited-by-osha-after-explosion-facing-in-penalties.html*

[3] (2013). J-15 Chief Architect: 3D printing used in developing new fighter jet [Web log post]. *Retrieved from http://www.3ders.org/articles/20130304-j-15-chief-architect-3d-printing-used-in-developing-new-fighter-jet.html*

[4] An, H., Rui, Z., Wang, R., & Zhang, Z. (2014). Research on Cutting-Temperature Field and Distribution of Heat Rates Among a Workpiece, Cutter, and Chip for High-Speed Cutting Based on Analytical and Numerical Methods. *Strength Of Materials*, *46*(2), 289-295.

[5] Griffey, J. (2014). The Types of 3-D Printing. *Library Technology Reports*,*50*(5), 8-12.

[6] Hull, C. W. (1986). U.S. Patent No. 4,575,330. *Washington, DC: U.S. Patent and Trademark Office.*

[7] Lifton, A.V., Lifton, G., & Simon, S. (2014). Options for additive rapid prototyping methods (3D printing) in MEMS technology. *Rapid Prototyping Journal*, *20*(5), 403-412.

[8] Diegel, O. (2011). Additive manufacturing: the new industrial revolution. *Lancaster University, Lancaster, UK.*

[9] Gibson, I., Rosen, D. W., & Stucker, B. (2010). Additive manufacturing technologies. *Springer, New York, NY.*

[10] Comb, J. W. (2003). U.S. Patent No. 6,547,995. *Washington, DC: U.S. Patent and Trademark Office.*

[11] Crump, S. S. (1992). U.S. Patent No. 5,121,329. *Washington, DC: U.S. Patent and Trademark Office.*

[12] Rodriguez, J. F., Thomas, J. P., & Renaud, J. E. (2000). Characterization of the mesostructure of fused-deposition acrylonitrile-butadiene-styrene materials. *Rapid Prototyping Journal*, *6*(3), 175-186.

[13] Bellehumeur, C., Li, L., Sun, Q., & Gu, P. (2004). Modeling of bond formation between polymer filaments in the fused deposition modeling process. *Journal of Manufacturing Processes*, *6*(2), 170-178.

[14] Paulsen, C., and Dempsey, K. (2014). Risk Management for Replication Devices. *National Institute for Standards and Technology*.

[15] Jones, R.A. & Horowitz, B.M. (2012). System-Aware Cyber Security Architecture, *Journal: Systems Engineering, Vol. 15 (2), pp. 224-240*

[16] Sood, A. K., Ohdar, R. K., & Mahapatra, S. S. (2010). Parametric appraisal of fused deposition modelling process using the grey Taguchi method. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *224*(1), 135-145.

[17] Paulsen, C. (2015). Direct Digital Manufacturing (DDM) Symposium. *National Institute for Standards and Technology*.

[18] Stratasys, Inc. (2012). uPrint® and uPrint® Plus uPrint® SE and uPrint® SE Plus Service Manual. *Stratasys, Inc., Eden Prairie, MN.*

[19] Stratasys, Inc. (2011). Characterization of Material Properties: Fortus ABS-M30. *White Paper, Stratasys, Inc., Eden Prairie, MN.*

[20] N. Turner, B., Strong, R., & A. Gold, S. (2014). A review of melt extrusion additive manufacturing processes: I. Process design and modeling. *Rapid Prototyping Journal*, *20*(3), 192-204.

[21] Rosenzweig, N., & Narkis, M. (1981). Sintering rheology of amorphous polymers. *Polymer Engineering & Science*, *21*(17), 1167-1170.

[22] Bellini, A., Guceri, S., & Bertoldi, M. (2004). Liquefier dynamics in fused deposition. *Journal of Manufacturing Science and Engineering*, *126*(2), 237-246.

[23] ASTM D638 – 10. (2010), Standard Test Method for Tensile Properties of Plastics, *ASTM International, West Conshohocken, PA*

[24] Ziemian, S., Okwara, M., & Ziemian, C. W. (2015). Tensile and fatigue behavior of layered acrylonitrile butadiene styrene. *Rapid Prototyping Journal*, *21*(3), 270-278.

[25] Leong, K. F., Cheah, C. M., & Chua, C. K. (2003). Solid freeform fabrication of three-dimensional scaffolds for engineering replacement tissues and organs.*Biomaterials*, *24*(13), 2363-2378.

[26] Stratasys, Inc. (2015). ABSplus-P430 Production-Grade Thermoplastic for Design Series 3D Printers. *Material Spec Sheet, Stratasys, Inc., Eden Prairie, MN.*

[27] Rodríguez, J. F., Thomas, J. P., & Renaud, J. E. (2001). Mechanical behavior of acrylonitrile butadiene styrene (ABS) fused deposition materials. Experimental investigation. *Rapid Prototyping Journal*, *7*(3), 148-158.

[28] Miller, R. J., & Derakhshan, M. (2002, March). The Invisalign System: Case report of apatient with deep bite, upper incisor flaring, and severe curve of spee. In *Seminars in Orthodontics* (Vol. 8, No. 1, pp. 43-50). WB Saunders.

[29] Tadjdeh, Y. (2014). Navy Beefs Up 3D Printing Efforts With New ‘Print the Fleet’ Program [Web log post]. *Retrieved From http://www.nationaldefensemagazine.org/archive/2014/October/Pages/NavyBeefsUp3DPrintingEffortsWithNewPrinttheFleetProgram.aspx*

[30] Jones, R.W., Mace, B.R., & Tham, M.T. (2002). “The Evolution of Mechanical Engineering Curricula: Mechatronics.” *International Conference on Education, Manchester, UK.*

[31] Rauwendaal, C. (1986). Polymer extrusion. *Hanser Gardner Pubs.*

[32] Gabay, J. (2012). Motor Sensing via Rotary Shaft Encoders Assures Safety and Control [Online article]. *Retrieved from http://www.digikey.com/en/articles/techzone/2012/aug/motion-sensing-via-rotary-shaft-encoders-assures-safety-and-control*

[33] N. Turner, B., & A. Gold, S. (2015). A review of melt extrusion additive manufacturing processes: II. Materials, dimensional accuracy, and surface roughness. *Rapid Prototyping Journal*, *21*(3), 250-261.

[34] Brebu, M., Bhaskar, T., Murai, K., Muto, A., Sakata, Y., & Uddin, M. A. (2004). Thermal degradation of PE and PS mixed with ABS-Br and debromination of pyrolysis oil by Fe- and Ca-based catalysts. *Polymer Degradation and Stability,84*(3), 459-467.

[35] US Sensor. (2006, revised 2009). Platinum RTD Sensor P/N PPG101A6. *PPG101A6 Datasheet.*

[36] Analog Devices. (1999). Monolithic Thermocouple Amplifiers with Cold Junction Compensation, Rev. C. *AD595 Datasheet*.

[37] Analog Devices. (2004). LC^{2}MOS Signal Conditioning ADC with RTD Excitation Currents, Rev G. *AD7711 Datasheet.*

Appendix A – Sentinel Assembly Code

```
{
This is the sentinel code developed for a Dimension uPrint 3D printer. This code monitors
the XYZ position of the printing head,
  the tip temperature of the model extruder, the status of the material sensor, and the
position of the extruder motor.
For proper initialization, this code must be run when the printer is physically at its
home position where X, Y, and Z are all zeroed.
The code below was originally all written in Spin, but slow refresh rates justified the
change to assembly code.

Cog Usage:

- Cog 0: Loads SD coordinates onto assembly cogs, then tells assembly cogs to run. Also
runs serial communication (Simple_Serial object) for debugging
- Cog 1: Runs SD card driver
- Cog 2: Assembly - Reads X and Y motor encoders
- Cog 3: Reserved for AC1
- Cog 4: Reserved for AC2
- Cog 5: Reads quadrature encoder on extruder motor (measures total position traveled by
extruder motor)
- Cog 6: Assembly - Reads Z axis encoder
- Cog 7: Assembly - Reads temperature data from RTD ADC chip
}
CON
        _clkmode = xtal1 + pll16x'Overclocked clock mode * crystal frequency = 100 MHz
        _xinfreq = 6_250_000

  ''Serial RTD ADC Pins
  A0=0
  SDATA=1
  DRDY=2
  RFS=3
  TFS=4
  SCLK=5

  ''Quadrature Z Encoder Pins
  zQuadA=6
  zQuadB=7
  ''Quadrature XY Encoder Pins
  xQuadA=8
  xQuadB=9
  yQuadA=10
  yQuadB=11
  ''Quadrature Extruder Encoder Pins
  Model=12
  Support=13
  ExtrQuadA=14
  ExtrQuadB=15
  ''SD Interface Pins
  DO=22
  SD_CLK=23
  SD_DI=24
  CS=25

  ''Simple_Serial Constants
  HOME=1
  TAB=9
  CR=13
  CLS=16


  ''RTD ADC Constants
  controlRegister=%1011_1000_1101_0001_0100_0110        'See pp. 9-10 of AD7711 datasheet

 {
    - RTD excitation current turned on
    - Background calibration mode enabled
    - Unipolar operation
    - PGA gain set to 64
    - Word Length set to 24 bits
    }

  ADCmax=16_777_215          'Max ADC unit is 2^24-1, which is 16,777,215
  Vdd=5                      'Supply voltage is 5V, internal reference voltage is 2.5V
  Aexc=200                   'Excitation current is 200 microamps


VAR

''CMB/SD Card Variables
long    m, n, DP, cmbByte, PM
long    addressPM, addressX
byte    lowerByte, upperByte

''Encoder Assembly Variables
long    xPosit, yPosit, zPosit, units

''Extruder Assembly Variables
long    printMaterial, periodCounts, roadWidth, ExtruderPosition

''Position Checking Assembly Variables
long    C1, C2, C3, flagA, flagB, DPNumber, end, loadedA, loadedB, expectedMaterial

''Failure Variables
long    extruderMisses, correctMaterial, correctExtruder

''RTD Variables
long    RTDtemp, ADCdata, RTDStack[100]

OBJ
  ss    : "Simple_Serial"
  sd    : "SD-MMC_FATEngine"
  fm    : "FloatMath"

PUB Main
''Initialize all variables and start running code on the parallel processors
Initialize

''Read in the first set of points into Hub RAM
flagA:=1
flagB:=1
DPNumber:=30
ReadPoints(DPNumber)
''Start the first assembly position checking cog (AC1)
```

```
coginit(3,@CheckPosition,0)
waitcnt(clkfreq/200+cnt)

''Uncomment this block of code to see the list of points contained in the CMB file.
{repeat
  repeat n from 0 to DPnumber-1
    ss.hex(long[@printMode][5*n],2)
    ss.tx(TAB)
    ss.dec(long[@x][5*n])
    ss.str(string(","))
    ss.dec(long[@y][5*n])
    ss.str(string(","))
    ss.dec(long[@z][5*n])
    ss.str(string(","))
    ss.dec(long[@width][5*n])
    ss.tx(CR)
  waitcnt(clkfreq+cnt)
  ReadPoints(DPNumber)

repeat                         'Output the SD data}

''Main Loop
repeat while end<>1
  ''Load CMB waypoints to Hub RAM and start AC1 with them. This only executes if AC2 is
running and Cog 3 isn't running AC1 code
  if flagA==1 AND LoadedA==0
    ReadPoints(DPNumber)
    coginit(3,@CheckPosition,0)
    waitcnt(clkfreq/200+cnt)

  ''Load CMB waypoints to Hub RAM and start AC2 with them. This only executes if AC1 is
running and Cog 4 isn't running AC2 code
  if flagB==1 AND LoadedB==0
    ReadPoints(DPNumber)
    coginit(4,@CheckPosition,0)
    waitcnt(clkfreq/200+cnt)

  ''Serial Terminal Output for Debugging
  waitcnt(clkfreq/20+cnt)
  ss.clearhome
  ss.dec(xPosit)
  ss.str(string(","))
  ss.dec(yPosit)
  ss.str(string(","))
  ss.dec(zPosit)
  ss.tx(CR)
  ss.dec(C1)
  ss.str(string(","))
  ss.dec(C2)
  ss.str(string(","))
  ss.dec(C3)
  ss.tx(CR)                '}
  ss.str(string("Extruder Position: "))
  ss.dec(ExtruderPosition)
  ss.tx(CR)
  ss.str(string("Expected Extruder State: "))
  ss.dec(roadWidth)
  ss.tx(CR)
  ss.str(string("RTD Temp: "))
  ss.dec(RTDTemp)
  ss.str(string(" C"))
  ss.tx(CR)

  'First output type
  ss.str(string("Current Print Mode: "))
  ss.hex(PM,2)
  ss.tx(CR)
  ss.str(string("FlagA: "))
  ss.dec(flagA)
  ss.tx(CR)
  ss.str(string("FlagB: "))
  ss.dec(flagB)
  ss.tx(CR)
  ss.str(string("loadedA: "))
  ss.dec(loadedA)
  ss.tx(CR)
  ss.str(string("loadedB: "))
  ss.dec(loadedB)
  ss.tx(CR)                '}
  ss.str(string("Print Material: "))
  ss.dec(printMaterial)
  ss.tx(CR)
  ss.str(string("Expected Material: "))
  ss.dec(expectedMaterial)
  ss.tx(CR)
  ss.str(string("Correct Material: "))
  ss.dec(correctMaterial)
  ss.tx(CR)          '}

''Finish by closing everything
EndCode

PUB ReadPoints(Points)
''Read the parameters from the FP-version of the CMB for the total number of points and
load them to Hub RAM
repeat n from 0 to Points-1
  ''Read the Print Mode
  cmbByte:=sd.readByte
  long[@printMode][5*n]:=cmbByte
  ''Search for the $FF endcode in the FP-version of the CMB
  if cmbByte==$FF
    quit

  ''Read in the X, Y, and Z coordinates, as well as the road width, for the current point
  lowerByte:=sd.readByte
  upperByte:=sd.readByte
  ''Store these quantities in Hub RAM at the addresses declared in the checking method in
the DAT section
  long[@x][5*n]:=lowerByte+upperByte<<8
  lowerByte:=sd.readByte
  upperByte:=sd.readByte
  long[@y][5*n]:=lowerByte+upperByte<<8
  lowerByte:=sd.readByte
  upperByte:=sd.readByte
  long[@z][5*n]:=lowerByte+upperByte<<8
  lowerByte:=sd.readByte
  upperByte:=sd.readByte
  long[@width][5*n]:=lowerByte+upperByte<<8
  waitcnt(clkfreq/200+cnt)

PUB Initialize
''Simple Serial Intialization
ss.init(31,30,38400)
ss.clearHome

''SD Initialization
sd.FATEngineStart(DO,SD_CLK,SD_DI,CS,-1,-1,-1,-1,-1)         'Start the SD card object
sd.mountPartition(0)                                   'Mount the SD card
''Change the file name to the currently-printing file. Use only FP-converted CMB files.
sd.openFile(string("TEMPFP.CMB"),"R")

''Assembly Variables (for writing/reading Hub RAM)
_DP:=@DP
_PM:=@PM
_xPosit:=@xPosit
_xPosition:=@xPosit
_flagA:=@flagA
_flagB:=@flagB
_DPNumber:=@DPNumber
_end:=@end
_loadedA:=@loadedA
_loadedB:=@loadedB
_yPosit:=@yPosit
_yPosition:=@yPosit
_printMaterial:=@printMaterial
_C1:=@C1
_C2:=@C2
_C3:=@C3
_zPosit:=@zPosit
_zPosition:=@zPosit
_roadWidth:=@roadWidth
_expectedMaterial:=@expectedMaterial
_extrPosit:=@ExtruderPosition
_corrMat:=@correctMaterial
_RTDTemp:=@RTDTemp

''Start all parallel codes except for AC1 and AC2
coginit(2,@QuadEnc,0)
coginit(5,@ExtrEncoder,0)
coginit(6,@ReadZEnc,0)
coginit(7,RetrieveRTDTemp,@RTDStack)


''Make sure that the code gets past initialization
waitcnt(clkfreq+cnt)
ss.clearHome
ss.str(string("Starting.."))
waitcnt(clkfreq+cnt)
ss.clearHome

PUB EndCode

waitcnt(clkfreq+cnt)
sd.unmountPartition                                    'Unmount the SD card

''Display the end code
repeat
```

```
  ss.clearHome
  ss.str(string("End of File Reached"))
  waitcnt(clkfreq+cnt)

PUB RetrieveRTDTemp
  ''Set the direction of the RTD ADC's serial pins
  dira[SDATA..TFS]:=%0011
  dira[AO]:=1
  dira[SCLK]:=1
  outa[AO]~~                  'Hold A0 high to read output register
  SetControlRegister          'Set the 24-bit control register on the ADC chip
  repeat
    ADCdata:=GetADCdata
    ''Scale RTDtemp into C and round to the nearest number
    RTDtemp:=fm.fRound(fm.fSub(fm.fMul(fm.fFloat(ADCdata),0.000030778676060),267.47))

PUB SetControlRegister | count
  outa[AO]~                                             'Hold A0 low to access the
control register
  dira[SDATA]~~                                         'Hold SDATA high before data
transfer
  outa[RFS..TFS]~~                                      'Hold RFS and TFS high before
data transfer
  outa[SCLK]~                                           'Hold the Serial Clock low before
data transfer
  outa[TFS]~                                            'Set TFS (transmit) low to begin
data write operation
  repeat count from 0 to 23                             'Iterate through all 24 bits
    outa[SDATA]:=(controlRegister>>(23-count))//2       'Shift data out to the ADC's
SDATA line one bit at a time, beginning with the MSB
    outa[SCLK]~~                                        'Pulse the Serial Clock once
    outa[SCLK]~
  outa[TFS]~~                                           'Return TFS high to end write
operation
  outa[AO]~~                                            'Return A0 high (to access output
register)
  dira[SDATA]~                                          'Set SDATA low after data
transfer

PUB GetADCdata : ADCBits
  outa[RFS..TFS]~~                                      'Set RFS and TFS high
  outa[SCLK]~                                           'Set SCLK low initially
  repeat until ina[DRDY]==0                             'Wait until Data Ready goes low
(output buffer ready)
  outa[RFS]~                                            'Set RFS low to begin data read
operation
  repeat 24                                             'Iterate through all 24 bits
    ADCbits:=(ADCbits<<1+ina[SDATA])                    'Read in the bits from the ADC's
serial output buffer, reversing the order of the bits
    outa[SCLK]~~                                        'Pulse the Serial Clock once
    outa[SCLK]~
  outa[RFS]~~                                           'Return RFS high to end read
operation

DAT

''This PASM code reads the A and B channel output signals of the extruder encoder and
''returns 1 if the extruder is turning CW, 0 if the extruder isn't turning, or -1 if the
extruder is turning CCW
```

```
              org
ExtrEncoder   mov       dira,#0                         'Set directions of ExtrQuadA and
ExtrQuadB to inputs
              mov       extrMask,#%11                   'Set the lower 2 bits of extrMask
high
              shl       extrMask,#ExtrQuadA             'Shift them over to the ExtrQuadA
and ExtrQuadB position
ExtrLoop      mov       extrIn,ina                      'Store entire ina register in
"input" register
              and       extrIn,extrMask                 'Chop off everything except
ina[ExtrQuadA..ExtrQuadB]
              shr       extrIn,#ExtrQuadA               'Put A and B values in the 0 and
1 position
              cmp       extrIn,extrPrev      wz         'If input=previous, set Z-flag
       if_z   jmp       #ExtrLoop

              cmp       extrIn,#%00       wz            'If input=%00, set Z-flag
       if_z   jmp       #ExtrCaseZero                      'Call "ExtrCaseZero"
subroutine
              cmp       extrIn,#%01       wz            'If input=%01, set Z-flag
       if_z   jmp       #ExtrCaseOne                       'Call "ExtrCaseOne" subroutine
              cmp       extrIn,#%10       wz            'If input=%10, set Z-flag
       if_z   jmp       #ExtrCaseTwo                       'Call "ExtrCaseTwo" subroutine
              cmp       extrIn,#%11       wz            'If input=%11, set Z-flag
       if_z   jmp       #ExtrCaseThree                     'Call "ExtrCaseThree"
subroutine


ExtrCaseZero  cmp       extrPrev,#%10     wz            'If input=%01, movement=CW, set
Z-flag
              mov       materialUse,ina
              mov       matMask,#1
              shl       matMask,#Support                'Make a mask to check the input
state of the support pin
              and       materialUse,matMask
              shl       materialUse,#Support            'Store the input state of Support
in "materialUse"
              cmp       materialUse,#1 wc               'Check for model material usage
if_nz_and_c   add       extrPos,#1                      'If movement=CW, add 1 to _xpos
if_z_and_c    sub       extrPos,#1                      'If movement=CCW, subtract 1 from
_xpos
              jmp       #StoreExtr

ExtrCaseOne   cmp       extrPrev,#%00     wz            'If input=%11, movement=CW, set
Z-flag
              mov       materialUse,ina
              mov       matMask,#1
              shl       matMask,#Support                'Make a mask to check the input
state of the support pin
              and       materialUse,matMask
              shl       materialUse,#Support            'Store the input state of Support
in "materialUse"
              cmp       materialUse,#1 wc               'Check for model material usage
if_nz_and_c   add       extrPos,#1                      'If movement=CW, add 1 to _xpos
if_z_and_c    sub       extrPos,#1                      'If movement=CCW, subtract 1 from
_xpos
              jmp       #StoreExtr
```

```
ExtrCaseTwo   cmp       extrPrev,#%11     wz            'If input=%00, movement=CW, set
Z-flag
              mov       materialUse,ina
              mov       matMask,#1
              shl       matMask,#Support                'Make a mask to check the input
state of the support pin
              and       materialUse,matMask
              shl       materialUse,#Support            'Store the input state of Support
in "materialUse"
              cmp       materialUse,#1 wc               'Check for model material usage
if_nz_and_c   add       extrPos,#1                      'If movement=CW, add 1 to _xpos
if_z_and_c    sub       extrPos,#1                      'If movement=CCW, subtract 1 from
_xpos
              jmp       #StoreExtr

ExtrCaseThree cmp       extrPrev,#%01     wz            'If input=%10, movement=CW, set
Z-flag
              mov       materialUse,ina
              mov       matMask,#1
              shl       matMask,#Support                'Make a mask to check the input
state of the support pin
              and       materialUse,matMask
              shl       materialUse,#Support            'Store the input state of Support
in "materialUse"
              cmp       materialUse,#1 wc               'Check for model material usage
if_nz_and_c   add       extrPos,#1                      'If movement=CW, add 1 to _xpos
if_z_and_c    sub       extrPos,#1                      'If movement=CCW, subtract 1 from
_xpos
              jmp       #StoreExtr

StoreExtr     mov       extrPrev,extrIn                 'Store previous input value as
"prev"
OutputExtr    wrlong    extrPos,_extrPosit
              jmp       #ExtrLoop

extrMask                long    0
matMask                 long    0
materialUse             long    0
extrIn                  long    0
extrPrev                long    0
extrPos                 long    0
_extrPosit              long    0
                        fit

''This PASM code reads the A and B channel output signals of the X and Y quadrature
encoders.
''  It writes the X and Y axial positions in scaled fixed-point format back to Hub RAM
              org
QuadEnc       mov       dira,#0                 'Set directions of xQuadA and xQuadB to
inputs
              mov       xMask,#%11              'Set the lower 2 bits of xMask high
              shl       xMask,#xQuadA           'Shift them over to the xQuadA and xQuadB
position
              mov       yMask,#%11              'Set the lower 2 bits of yMask high
              shl       yMask,#yQuadA           'Shift them over to the yQuadA and yQuadB
position
LoopX         mov       xIn,ina                 'Store entire ina register in "input"
register
```

```
              and       xIn,xMask                     'Chop off everything except
ina[xQuadA..xQuadB]
              shr       xIn,#xQuadA                   'Put A and B values in the 0 and 1
position
              cmp       xIn,xPrev      wz       'If input=previous, set Z-flag
       if_z   jmp       #LoopY

              cmp       xIn,#%00       wz       'If input=%00, set Z-flag
       if_z   jmp       #caseZeroX                      'Call "caseZeroX" subroutine
              cmp       xIn,#%01       wz       'If input=%01, set Z-flag
       if_z   jmp       #caseOneX                       'Call "caseOneX" subroutine
              cmp       xIn,#%10       wz       'If input=%10, set Z-flag
       if_z   jmp       #caseTwoX                       'Call "caseTwoX" subroutine
              cmp       xIn,#%11       wz       'If input=%11, set Z-flag
       if_z   jmp       #caseThreeX                     'Call "caseThreeX" subroutine


caseZeroX     cmp       xPrev,#%10     wz       'If input=%01, movement=CW, set Z-flag
if_z          add       xPos,#1                 'If movement=CW, add 1 to _xpos
if_nz         sub       xPos,#1                 'If movement=CCW, subtract 1 from _xpos
              jmp       #StoreX

caseOneX      cmp       xPrev,#%00     wz       'If input=%11, movement=CW, set Z-flag
if_z          add       xPos,#1                 'If movement=CW, add 1 to _xpos
if_nz         sub       xPos,#1                 'If movement=CCW, subtract 1 from _xpos
              jmp       #StoreX

caseTwoX      cmp       xPrev,#%11     wz       'If input=%00, movement=CW, set Z-flag
if_z          add       xPos,#1                 'If movement=CW, add 1 to _xpos
if_nz         sub       xPos,#1                 'If movement=CCW, subtract 1 from _xpos
              jmp       #StoreX

caseThreeX    cmp       xPrev,#%01     wz       'If input=%10, movement=CW, set Z-flag
if_z          add       xPos,#1                 'If movement=CW, add 1 to _xpos
if_nz         sub       xPos,#1                 'If movement=CCW, subtract 1 from _xpos
              jmp       #StoreX

StoreX        mov       xPrev,xIn           'Store previous input value as "xPrev"

          ''Multiply xPos by the numerator. This was hard-coded to save time
          ''  (old algorithm iterated and took forever).This method uses the
          ''   binary-decomposed version of the numerator, as shown in the
          ''   comment below by the bit-shifting section of the code.
          mov     t0,xPos         'Store xPos in 6 temporary variables for bit shifting
          mov     t1,xPos
          mov     t2,xPos
          mov     t3,xPos
          mov     t4,xPos
          mov     t5,xPos         '              0       1       2       3      4      5
          shl     t0,#14          'x * 19_356 = x<<14 + x<<11 + x<<10 - x<<6 - x<<5 -
x<<2
          shl     t1,#11
          shl     t2,#10
          shl     t3,#6
          shl     t4,#5
          shl     t5,#2
          add     t0,t1           'Execute the above equation
          add     t0,t2
          sub     t0,t3
          sub     t0,t4
          sub     t0,t5
          mov     xScaled,t0

          ''Divide xPosition by the denominator (Also hard-coded)
          mov     quotient,xScaled
          cmps    quotient,#0 wc
          'Convert to Two's Complement Notation in the Case of a Negative Number (C=1)
if_c      mov     quotient2sC,quotient    'Copy the negative quotient into quotient2sC
if_c      xor     quotient2sC,LargestNum  'Flip all the bits of quotient2sC
if_c      add     quotient2sC,#1          'Add 1 to quotient2sC
if_c      mov     quotient,quotient2sC    'Store the positive form of the quotient back
in "quotient"
          shr     quotient,#14            'Divide by 16_384
if_c      xor     quotient,LargestNum     'Flip all the bits to convert back to a
negative number
if_c      add     quotient,#1             'Add 1 to finish the 2's Complement conversion
          mov     xScaled,quotient
          wrlong  xScaled,_xPosit         'Write the scaled xPosition to the xPosit
variable }



''Retrieve the Y Position from the Y Encoder
LoopY         mov       yIn, ina
              and       yIn,yMask               'Chop off everything except
ina[yQuadA..yQuadB]
              shr       yIn,#yQuadA             'Put A and B values in the 0 and 1
position
              cmp       yIn,yPrev      wz       'If input=previous, set Z-flag
       if_z   jmp       #LoopX

              cmp       yIn,#%00       wz       'If input=%00, set Z-flag
       if_z   jmp       #caseZeroY                      'Call "caseZeroX" subroutine
              cmp       yIn,#%01       wz       'If input=%01, set Z-flag
       if_z   jmp       #caseOneY                       'Call "caseOneX" subroutine
              cmp       yIn,#%10       wz       'If input=%10, set Z-flag
       if_z   jmp       #caseTwoY                       'Call "caseTwoX" subroutine
              cmp       yIn,#%11       wz       'If input=%11, set Z-flag
       if_z   jmp       #caseThreeY                     'Call "caseThreeX" subroutine


caseZeroY     cmp       yPrev,#%10     wz       'If input=%01, movement=CW, set Z-flag
if_z          sub       yPos,#1                 'If movement=CW, add 1 to _xpos
if_nz         add       yPos,#1                 'If movement=CCW, subtract 1 from _xpos
              jmp       #StoreY

caseOneY      cmp       yPrev,#%00     wz       'If input=%11, movement=CW, set Z-flag
if_z          sub       yPos,#1                 'If movement=CW, add 1 to _xpos
if_nz         add       yPos,#1                 'If movement=CCW, subtract 1 from _xpos
              jmp       #StoreY

caseTwoY      cmp       yPrev,#%11     wz       'If input=%00, movement=CW, set Z-flag
if_z          sub       yPos,#1                 'If movement=CW, add 1 to _xpos
if_nz         add       yPos,#1                 'If movement=CCW, subtract 1 from _xpos
              jmp       #StoreY

caseThreeY    cmp       yPrev,#%01     wz       'If input=%10, movement=CW, set Z-flag
if_z          sub       yPos,#1                 'If movement=CW, add 1 to _xpos
```

```
if_nz         add       yPos,#1                 'If movement=CCW, subtract 1 from _xpos
              jmp       #StoreY

StoreY        mov       yPrev,yIn               'Store previous input value as "prev"


          ''Multiply yPos by the numerator (Also hard-coded)
          mov     answer,#0
          mov     t0,yPos         'Store xPos in 6 temporary variables for bit shifting
          mov     t1,yPos
          mov     t2,yPos        '             0       1      2
          shl     t0,#13          'x * 7_743 = x<<13 - x<<9 + x<<6 - x<<0
          shl     t1,#9
          shl     t2,#6
          sub     t0,t1           'Execute the above equation
          add     t0,t2
          sub     t0,yPos
          mov     yScaled,t0

          ''Divide xPosition by the denominator (Also hard-coded)
          mov     quotient,yScaled
          cmps    quotient,#0 wc
          'Convert to Two's Complement Notation in the Case of a Negative Number (C=1)
if_c      mov     quotient2sC,quotient    'Copy the negative quotient into quotient2sC
if_c      xor     quotient2sC,LargestNum  'Flip all the bits of quotient2sC
if_c      add     quotient2sC,#1          'Add 1 to quotient2sC
if_c      mov     quotient,quotient2sC    'Store the positive form of the quotient back
in "quotient"
          shr     quotient,#14            'Divide by 16_384
if_c      xor     quotient,LargestNum     'Flip all the bits to convert back to a
negative number
if_c      add     quotient,#1             'Add 1 to finish the 2's Complement conversion
          mov     yScaled,quotient
          wrlong  yScaled,_yPosit         'Write the scaled y position to the yPosit
variable }


          jmp       #LoopX              'Restart loop at beginning

''Quadrature Variables - X Axis
xMask                   long    0
xIn                     long    0
xPrev                   long    0
xPos                    long    0
_xPosit                 long    0

''Quadrature Variables - Y Axis
yMask                   long    0
yIn                     long    0
yPrev                   long    0
yPos                    long    0
_yPosit                 long    0

''Multiplication/Division Variables
xScaled                 long    0
yScaled                 long    0
t0                      long    0
t1                      long    0
t2                      long    0
```

```
t3                      long    0
t4                      long    0
t5                      long    0
'xNumerator              long    19_356
'yNumerator              long    7_743
'denominator             long    16_384    (Just for reference)
a                       long    0
b                       long    0
bAND                    long    0
answer                  long    0
quotient                long    0
minusOne                long    -1
quotient2sC             long    0
LargestNum              long    %1111_1111_1111_1111_1111_1111_1111_1111
                        fit


''This PASM code checks to see off move coordinates based on the position variables
stored in Hub RAM from the encoders.

                        org 0
                        '' Check which cog is currently in use
CheckPosition           cogID   cogNum
                        mov     dataPoint,#0

                        ''Set loadedA/loadedB (1 while running)
                        cmp     cogNum,#4 wz
              if_nz     wrlong  high,_loadedA
              if_z      wrlong  high,_loadedB

                        ''Retrieve the flag variables from Hub RAM
WaitFlag                rdlong  flgA,_flagA
                        rdlong  flgB,_flagB
                        mov     flag,flgB
                        and     flag,flgA

                        ''Wait for the flag from cogs 3 and 4 to go high (neither cog
running)
                        cmp     flag,#1 wz
              if_nz     jmp     #WaitFlag

                        ''Retrieve the total number of dataPoints for this run
                        rdlong  DPNum,_DPNumber

                        ''Operate the flags
                        cmp     cogNum,#4 wz
                        cmp     dataPoint,DPnum wc
        if_c_and_z      mov     flgB,#0
        if_nc_and_z     mov     flgB,#1
        if_c_and_nz     mov     flgA,#0
        if_nc_and_nz    mov     flgA,#1
                        wrlong  flgA,_flagA
                        wrlong  flgB,_flagB


PosLoop                 ''Calculate the position that we're waiting for
                        ''Store the current X value in xUp and xDown
                        ''  and likewise for Y and Z
```

```
XSourceUp               mov     xUp,x                   'Find the acceptable ranges for x
and y
XSourceDown             mov     xDown,x
YSourceUp               mov     yUp,y
YSourceDown             mov     yDown,y
ZSourceUp               mov     zUp,z
ZSourceDown             mov     zDown,z
PrintSource             mov     pMode,printMode         'See if the endCode ($FF) is
stored in the printMode variable
WidthSource             mov     rWidth,width

                        '' Check to see if the extruder should turn (road width in the FP
CMB)
                        cmp     rWidth,#0 wz
              if_z      mov     extrTurnOff,#0
              if_nz     mov     extrTurnOff,#1
                        wrlong  extrTurnOff,_roadWidth      '}

                        ''Write the current printMode and waypoint coordinates back to
Hub RAM
                        wrlong  pMode,_PM
                        cmp     pMode,#255 wz
        if_z            wrlong  high,_end

                        ''Check to see if the support head is being used (Not used for
Y!)
                        cmp     pMode,#$6A wc
              if_nc     add     xUp,XYOffset          'Offset X positions 798 mils
              if_nc     add     xDown,XYOffset
              if_nc     mov     expMat,#1             'Use 1 to represent the support
material
              if_c      mov     expMat,#0             'Use 0 to represent the model
material
              if_c      subs    zUp,partOffset        'Subtract 9 mils from z for model
material
              if_c      subs    zDown,partOffset      'Ditto for the lower limit

                        ''Store the current waypoint as (C1, C2, C3)
                        wrlong  xUp,_C1
                        wrlong  yUp,_C2
                        wrlong  zUp,_C3
                        wrlong  expMat,_expectedMaterial


                        ''Create ranges on waypoints
                        subs    xDown,acceptRange
                        adds    xUp,acceptRange
                        subs    yDown,acceptRange
                        adds    yUp,acceptRange
                        subs    zDown,zTolerance
                        adds    zUp,zTolerance

CheckLoop               ''This section checks the X, Y, Z, Material, and Extrusion Rate
                        'Retrieve the xPosition from Hub RAM
                        rdlong  xPosition,_xPosition
                        rdlong  yPosition,_yPosition
                        rdlong  zPosition,_zPosition
                        'Start by setting locFound
                        mov     locFound,#1
                        ''X Checking
                        cmps    xUp,xPosition wc        'Check if xUp is below xPosition
              if_c      mov     locFound,#0             'Clear locFound if the xPosition
falls out of range
                        cmps    xPosition,xDown wc      'Check if xPosition is below
xDown
              if_c      mov     locFound,#0             'Clear locFound if xPosition is
below xDown
                        ''Y Checking
                        cmps    yUp,yPosition wc
              if_c      mov     locFound,#0
                        cmps    yPosition,yDown wc
              if_c      mov     locFound,#0
                        ''Z Checking
                        cmps    zUp,zPosition wc
              if_c      mov     locFound,#0
                        cmps    zPosition,zDown wc
              if_c      mov     locFound,#0           '}
                        ''Material Checking
                        mov     material,#0
                        mov     inMask,#%11             'Store %11 in the lower nibble of
diraMask
                        shl     inMask,#Model           'Shift bits over to the Model and
Support Material input pins
                        mov     material,ina            'Store the entire ina register as
"input"
                        and     material,inMask         'Retain only the input states of
Model and Support
                        shr     material,#Model         'Move the input states back to
the LSB
                        cmp     material, #%10 wz
                        mov     material,#0             'Set material to model
              if_z      mov     material,#1             'If the support channel is high,
set the material to support
                        wrlong  material,_printMaterial
                        cmp     material,expMat wz
              if_z      mov     corrMat,#1
              if_nz     mov     corrMat,#0
                        and     locFound, corrMat
                        wrlong  corrMat, _corrMat       'Write the check status of the
material sensor
                        ''Temperature Checking
                        rdlong  tipTemp,_RTDTemp        'Retrieve the model tip
temperature from Hub RAM
                        cmp     material, #0 wz         'Check to see if model material
is printing currently
                        cmp     tipTemp,#200 wc         'Check to see if the current tip
temperature is below 200
        if_z_and_c      mov     locFound,#0             'If the model tip is printing
below 200 C, then reject point


                        ''Loop
                        cmp     locFound,#1 wz          'See if the current xPosition
falls into the stored x range
              if_nz     jmp     #CheckLoop

        if_z            add     dataPoint,#1            'Increment the dataPoint
        if_z            mov     _x,#x
        if_z            add     _x,dataPoint            'Increment the x address by
dataPoint*4
        if_z            add     _x,dataPoint
        if_z            add     _x,dataPoint
        if_z            add     _x,dataPoint
        if_z            add     _x,dataPoint
        if_z            movs    XSourceUp,_x            'Update the address that xUp and
xDown are copied from
        if_z            movs    XSourceDown,_x
        if_z            mov     _y,_x
        if_z            add     _y,#1                   'The nth y address is 1 long
after the nth x address
        if_z            movs    YSourceUp,_y
        if_z            movs    YSourceDown,_y
        if_z            mov     _z,_x
        if_z            add     _z,#2
        if_z            movs    ZSourceUp,_z
        if_z            movs    ZSourceDown,_z
        if_z            mov     PMaddress,_x
        if_z            sub     PMaddress,#2             'The nth PrintMode address is 2
longs before the nth x address
        if_z            movs    PrintSource,PMaddress    'Update the address that pMode
is copied from
        if_z            mov     _width,_x
        if_z            sub     _width,#1
        if_z            movs    WidthSource, _width


                        ''Check to see if the DP num is exceeded before reading an FF end
code
                        cmp     cogNum,#4 wz
                        cmp     dataPoint,DPnum wc
        if_c            jmp     #PosLoop                'Loop back to the beginning
                        ''Write the flags (0 for the cog that is running code, 1 for the
one that's not)
        if_c_and_z      mov     flgB,#0
        if_nc_and_z     mov     flgB,#1
        if_c_and_nz     mov     flgA,#0
        if_nc_and_nz    mov     flgA,#1
                        wrlong  flgA,_flagA
                        wrlong  flgB,_flagB

        if_nz           wrlong  zero,_loadedA           'Clear the loaded variables
        if_z            wrlong  zero,_loadedB
                        cogstop cogNum

'Position Checking Variables
xPosition               long    0
yPosition               long    0
zPosition               long    0
_C1                     long    0
_C2                     long    0
_C3                     long    0
_ExtruderControl        long    0
locFound                long    0
_end                    long    0
_loadedA                long    0
_loadedB                long    0
high                    long    1
```

```
zero                    long    0
flag                    long    0
flgA                    long    0
XYOffset                long    798
flgB                    long    0
_flagA                  long    0
_flagB                  long    0
cogNum                  long    0
DPNum                   long    30
_DPNumber               long    0
_xPosition              long    0
_yPosition              long    0
_zPosition              long    0
_PM                     long    0
_width                  long    0
pMode                   long    0
rWidth                  long    0
_roadWidth              long    0
PMaddress               long    0
inMask                  long    0
material                long    0
corrMat                 long    0
_corrMat                long    0
extrStatus              long    0
_printMaterial          long    0
expMat                  long    0
_expectedMaterial       long    0
extrTurnOff             long    0
eighty                  long    80_000
sixty                   long    60_000
forty                   long    40_000
twenty                  long    20_000
ten                     long    10_000
acceptRange             long    10
zTolerance              long    7
tipTemp                 long    0
_RTDTemp                long    0
dataPoint               long    0
_DP                     long    0
xUp                     long    0
xDown                   long    0
yUp                     long    0
yDown                   long    0
zUp                     long    0
zDown                   long    0
partOffset              long    9
suppOffset              long    2
_x                      long    0
_y                      long    0
_z                      long    0
printMode               long    0
width                   long    0
x                       long    0
y                       long    0
z                       long
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,{

}0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,{
}0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,{

}0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,{


}0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,{

}0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,{    Reserve 330 longs for
printMode, x, y, and z storage.

}0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,{    This lets us hold
330/4=82 points maximum in Cog RAM.

Thus, AC1 and AC2 can each check up to 82 points

}0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,{



}0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                        fit


''This PASM code retrieves the zPosition from the Z encoder

              org
ReadZEnc      mov       dira,#0            'Set directions of zQuadA and zQuadB to zIns
              mov       zMask,#%11         'Set the lower 2 bits of zMask high
              shl       zMask,#zQuadA      'Shift them over to the zQuadA and zQuadB
position
Loop          mov       zIn,ina            'Store entire ina register in "zIn" register
              and       zIn,zMask          'Chop off everything except ina[zQuadA..zQuadB]
              shr       zIn,#zQuadA        'Shift the zIn zQuadA number of bits to the
right to put A and B values in the 0 and 1 position
              cmp       zIn,zPrev     wz   'If zIn=previous, set Z-flag
       if_z   jmp       #Loop

              cmp       zIn,#%00     wz   'If zIn=%00, set Z-flag
       if_z   jmp       #caseZeroZ               'Call "caseZeroZ" subroutine
              cmp       zIn,#%01     wz   'If zIn=%01, set Z-flag
       if_z   jmp       #caseOneZ                'Call "caseOneZ" subroutine
              cmp       zIn,#%10     wz   'If zIn=%10, set Z-flag
       if_z   jmp       #caseTwoZ                'Call "caseTwoZ" subroutine
              cmp       zIn,#%11     wz   'If zIn=%11, set Z-flag
       if_z   jmp       #caseThreeZ              'Call "caseThreeZ" subroutine


caseZeroZ     cmp       zPrev,#%10     wz  'If zIn=%01, movement=CW, set Z-flag
if_z          adds      zPos,#1            'If movement=CW, add 1 to zPos
if_nz         subs      zPos,#1            'If movement=CCW, subtract 1 from zPos
              jmp       #StoreZ

caseOneZ      cmp       zPrev,#%00     wz  'If zIn=%11, movement=CW, set Z-flag
if_z          adds      zPos,#1            'If movement=CW, add 1 to zPos
if_nz         subs      zPos,#1            'If movement=CCW, subtract 1 from zPos
              jmp       #StoreZ

caseTwoZ      cmp       zPrev,#%11     wz  'If zIn=%00, movement=CW, set Z-flag
```

```
if_z          adds      zPos,#1            'If movement=CW, add 1 to zPos
if_nz         subs      zPos,#1            'If movement=CCW, subtract 1 from zPos
              jmp       #StoreZ

caseThreeZ    cmp       zPrev,#%01     wz     'If zIn=%10, movement=CW, set Z-flag
if_z          adds      zPos,#1            'If movement=CW, add 1 to zPos
if_nz         subs      zPos,#1            'If movement=CCW, subtract 1 from zPos
              jmp       #StoreZ

StoreZ        mov       zPrev,zIn           'Store previous zIn value as "zPrev"

           ''Multiply xPos by the numerator. This was hard-coded to save time
           ''  (old algorithm iterated and took forever). This method uses the
           ''  binary-decomposed version of the numerator, as shown in the comment
           ''  below by the bit-shifting section of the code.
          mov     temp0,zPos         'Store xPos in 6 temporary variables for bit
shifting
          mov     temp1,zPos
          mov     temp2,zPos
          mov     temp3,zPos
          mov     temp4,zPos
          mov     temp5,zPos
          mov     temp6,zPos         '             temp0   temp1   temp2  temp3  temp4
temp5  temp6
          shl     temp0,#13          'x * 12_881 = x<<13 + x<<12 + x<<9 + x<<6 + x<<3 +
x<<1 + x<<0
          shl     temp1,#12
          shl     temp2,#9
          shl     temp3,#6
          shl     temp4,#3
          shl     temp5,#1
          add     temp0,temp1           'Execute the above equation
          add     temp0,temp2
          add     temp0,temp3
          add     temp0,temp4
          add     temp0,temp5
          add     temp0,temp6
          mov     zScaled,temp0

          ''Divide xPosition by the denominator (Also hard-coded)
          mov     zQuotient,zScaled
          cmps    zQuotient,#0 wc
          ''Convert to Two's Complement Notation in the Case of a Negative Number (C=1)
if_c      mov     zQuotient2sC,zQuotient   'Copy the negative quotient into
quotientemp2sC
if_c      xor     zQuotient2sC,zLargestNum 'Flip all the bits of quotientemp2sC
if_c      add     zQuotient2sC,#1          'Add 1 to quotientemp2sC
if_c      mov     zQuotient,zQuotient2sC   'Store the positive form of the quotient back
in "quotient"
          shr     zQuotient,#14            'Divide by 16_384
if_c      xor     zQuotient,zLargestNum    'Flip all the bits to convert back to a
negative number
if_c      add     zQuotient,#1             'Add 1 to finish the 2's Complement conversion
          mov     zScaled,zQuotient
          wrlong  zScaled,_zPosit          'Write the scaled xPosition to the xPosit
variable }

          jmp     #Loop                    'Restart loop at beginning
```

```
''Quadrature Variables
zMask                   long    0
zIn                     long    0
zPrev                   long    0
zPos                    long    0
_zPosit                 long    0

''Multiplication/Division Variables
zScaled                 long    1023
zQuotient               long    0
zQuotient2sC            long    0
zLargestNum             long    %1111_1111_1111_1111_1111_1111_1111_1111
temp0                   long    0
temp1                   long    0
temp2                   long    0
temp3                   long    0
temp4                   long    0
temp5                   long    0
temp6                   long    0
                        fit
```

Appendix B – MATLAB Code

Appendix B.1 - Retrieves Data from Tensile Test Files

```
% This program retrieves the data from the .csv files created by the
% Instron computer. They must be formatted like the 1-28-15 test results

%Reading from .csv files with header information and 3 columns:
%--Extension is displacement of the crosshead, in mm (not used)
%--Displacement is the displacement of the laser tags, in mm (used for
    %strain calculation)
%--Load is the load cell reading, in Newtons (used for stress calculation)

clear all
clc

fileNumber=input('How many files do you have: ');
for i=1:fileNumber
    % Prefix for filename
    fileName=input('Enter the file prefix: ','s');

    % Number of specimens in the Sample
    N=6;

    %Populate Stress and Strain matrices with non-numbers
    eval([fileName,'.stress=NaN(2000,N);']);
    eval([fileName,'.strain=NaN(2000,N);']);

    for specimen=1:N
        %identify file with tension test data

fid=fopen([fileName,'.is_tens_RawData/Specimen_RawData_',num2str(specimen),'.
csv']);

        %save as gage length, in mm
        %(used to convert laser disp. into strain)
        temp(specimen)=textscan(fid,'%*s %q',1,'delimiter',',');

eval([fileName,'.gl(specimen)=str2num(cell2mat(temp{1,specimen}));']);


        %save as specimen width (in mm, converting to m)
        temp(specimen)=textscan(fid,'%*s %q %*s',1,'delimiter',',');

eval([fileName,'.w(specimen)=str2num(cell2mat(temp{1,specimen}))/1000;']);
        % save as specimen thickness (in mm, converting to m)
        temp(specimen)=textscan(fid,'%*s %q %*s',1,'delimiter',',');

eval([fileName,'.t(specimen)=str2num(cell2mat(temp{1,specimen}))/1000;']);
        %calculate cross-sectional area, in m^2
```

```
eval([fileName,'.A(specimen)=',fileName,'.w(specimen)*',fileName,'.t(specimen
);']);

        %extract tension test data from columns
        temp=textscan(fid,'%*q %q %q','delimiter',',','HeaderLines',6);
        for j=1:length(temp{1,1})

eval([fileName,'.displacement(j,specimen)=str2num(temp{1,1}{j,1});']); %
laser displacement in mm
            eval([fileName,'.load(j,specimen)=str2num(temp{1,2}{j,1});']);
% load in N
            eval([fileName,'.A(specimen)=3.83015E-05']);
            if strcmp(fileName,'SHD_XY')
                eval([fileName,'.A(specimen)=3.1315E-05']);
            end

eval([fileName,'.stress(j,specimen)=',fileName,'.load(j,specimen)/',fileName,
'.A(specimen)/1000000;']);

eval([fileName,'.strain(j,specimen)=',fileName,'.displacement(j,specimen)/',f
ileName,'.gl(specimen);']);
        end
    end
end
clear temp

% You must manually save the .mat file after running this code by selecting
% the structures for each file, right clicking, and then clicking "Save As"
```

Appendix B.2 - Plots Tensile Test Data Stored in the Workspace

```
% This program plots the tensile test data in the Tensile_Test_Data.mat
% file. The program must be run for each structure (attack).

clear all
clc

% Start by importing the Tensile_Test_Data.mat file
load Tensile_Test_Data.mat

%% Then plot the stress/strain curves
clc

N=6;
fileName='Solid_XZ_with_Notch';
linespec='''--k''';
hold on
for specimen=[1,2,3,5,6]%1:N

eval(['plot(',fileName,'.strain(:,specimen),',fileName,'.stress(:,specimen));
']);

%eval(['h.',fileName,'(i)=plot(strain.',color,'{i},stress.',color,'{i},',line
spec,')'])
end

% Now label the plots

legend('Specimen 1','Specimen 2','Specimen 3','Specimen 4','Specimen 5')
xlim([0,0.085])
ylim([0,30])
title(['Tension Testing Results for Hacked Solid XZ Specimens with a Seam'])
grid on
xlabel 'Strain'
ylabel 'Stress, MPa'
```

Appendix B.3 – Calculates Statistical Mean and Standard Deviation of Load Tests

```
% This program calculates the mean and deviation for a batch (attack). It
must be run once per batch

% Import all of the data batch files
for i=1:2
    eval(['load Batch_',num2str(i),'.mat'])
end

%% Calculate the statistical mean and standard deviation for each batch
clc

%Disregard one specimen because of noisy tensile test results
N=5;

M=NaN(N,1);

   for specimen=1:N
        M(specimen)=max(Solid_XZ.stress(:,specimen));
        UTS_control=mean(M);
        stDev_control=std(M);
    end
```