2016-01-01

Creating Multi-Functional G-Code For Multi-Process Additive Manufacturing

Efrain Aguilera Jr
University of Texas at El Paso, efrainaguilera1@gmail.com

CREATING MULTI-FUNCTIONAL G-CODE FOR MULTI-PROCESS ADDITIVE MANUFACTURING

EFRAIN AGUILERA JR

Master's Program Electrical Engineering

APPROVED:

Eric MacDonald, Ph.D., Chair

Rodrigo Romero, Ph.D.

David Roberson, Ph.D.

Charles Ambler, Ph.D.
Dean of the Graduate School

Dedication

I would like to dedicate this work to my mother and father, for their unconditional support. To my brother Carlos who pushed me to do my best.

SOFTWARE INCORPORATION OF MULITPLE TECHNOLOGIES INTO ADDITIVE MANUFACTURING SYSTEMS

by

EFRAIN AGUILERA JR, B.S.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

December 2016

Acknowledgements

I would like to thank Dr. Eric MacDonald, my advisor during my undergraduate and graduate studies. I am grateful for his constructive criticism and the opportunity he gave me to work at the W.M. Keck Center for 3D Innovation. I would also like to thank Dr. Ryan Wicker for his support and wise words during my research at the center. David Espalin Jr has been a great mentor, with his vast experience and guidance at problem solving and keeping the team on track.

I would like to extend my sincere appreciation to Mr. Callum Bailey who joined the team and helped enhance not only the project, but also our lives with his enchanting British accent. I would like to also thank Jake Lasley and Mike Licerio for their enhancements implemented into the GUI. Additionally I would like to thank Jose Motta and Alfonso Fernandez who designed the hardware. The MacDonald group composed of Donna, Issac, Jaime, Antonio, Luis Jr, Jazmin, Kelly, Matt, Luis C, and Luis B who helped get me through the long hard days spent inside the Keck Center.

I am indebted with the Keck Center for the great working environment provided that caters to the working student and every student that has gone through it while I have been there.

Lastly, I would like to thank America Makes, with their pioneering vision of making Additive Manufacturing technologies mainstream they have provided a grant funding this thesis.

Abstract

Additive manufacturing (AM) started over thirty years ago and with it a manufacturing revolution that moves industrial production into the personal home. With recent interest shifting into multi-functional parts fabricated through AM technologies, unified systems are being developed. Merging different manufacturing technologies into one single machine is a challenge but undergoing research has shown promise in the development of multi-functional systems. Concurrent work is being done in the software, automation, and hardware aspect of multi-functional systems. An effort to use industry compatible Computer Aided Design (CAD) software to design multi-functional parts including circuits, micro-machining, and foil embedding then exporting and processing automatically to create a hybrid g-code file is being done. The multi-functional g-code files include all the necessary information to create AM multi-functional parts without human intervention in the unified systems.

One advantage of AM is the ability to quickly prototype, extending this advantage to multi-functional parts means that quick multi-functional prototypes can be produce. Modern AM process are also capable of producing end use parts that are ready for commercial use, this leads to the possibility of creating end user parts with multi-functionality. This thesis explores the challenges and approaches to developing software that interfaces and processes multi-functional CADs and creates files for direct use in multi-functional AM machines.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 History

Recently there has been an explosion in the Additive Manufacturing (AM) world despite the technology being out for over 30 years (Gibson, Rosen & Stucker, 2010). As AM becomes more mainstream the usability has increased tremendously, once regarded as a rapid prototyping technology it is now implemented in end use parts. According to Wohlers report (Wohlers, 2016) Figure 1.1 shows the revenue of AM parts and services, showing an exponential growth in the last couple of years. As the technology has matured in the mechanical realm thoughts described in Medina *et al.* (2016) want to take advantage of AM and incorporate multiple technologies to create 3D printed electronic parts.

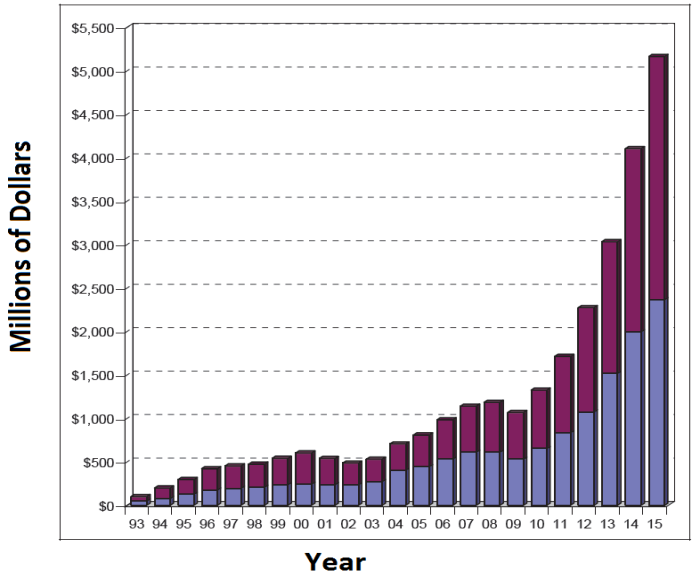


Figure 1.1 AM parts and Services (Wholers Report, 2016)

A multi-functional part is described as a part that has more than a physical use, it can include embedded circuits or foreign objects such as mesh for enhanced physical properties (Espalin, 2014). Although there are multiple ways of integrating circuits into different types of AM, technologies as portrayed in Aguilera *et al.* (2013) have a clear advantage in using copper

wire and thermoplastics for high power applications. Despite the advantages in embedding wire into thermoplastics, depositing conductive ink is very popular as seen in Lopes *et al.* (2006) and it continues to be a viable process for specific devices as depicted in Espalin *et al.* (2014). Stereolithography (SL) parts like that shown in Figure 1.2 with conductive ink was one of the first reported methods of both a stop and place process for embedding foreign items and electronic components inside an AM part.

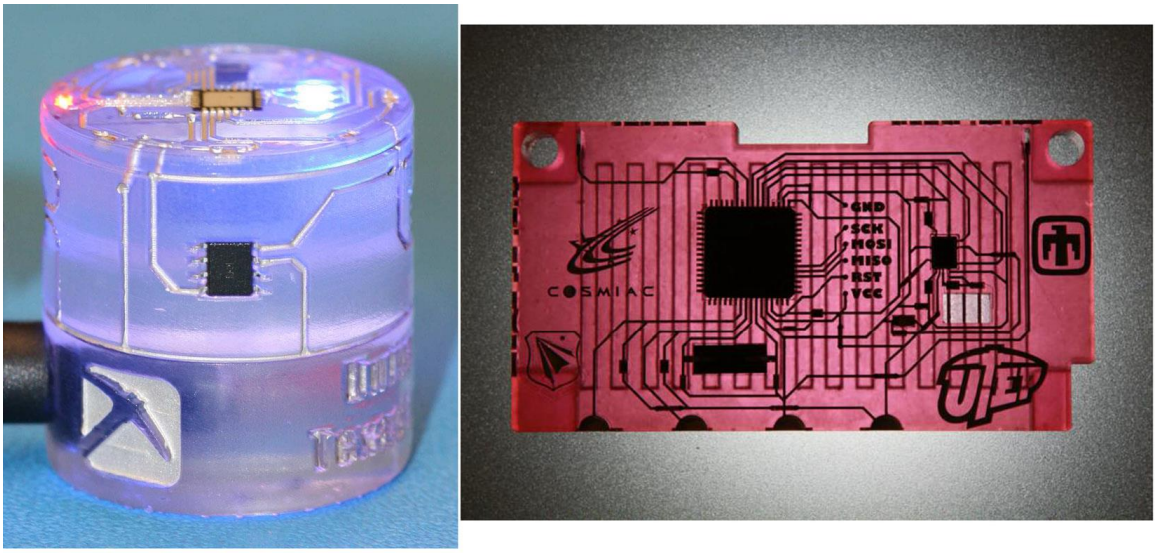


Figure 1.2 Left: Magnetometer, Right: CubeSat part (Espalin, 2014)

Due to the nature of the processes, material extrusion is a preferred solution for multi-functional parts. Material extrusion offers a larger selection of materials in which many are commercially available and tested in the real world including space applications (Nasa, 2016). Further, material extrusion is a cleaner process and is a popular AM process in the open source community. Despite its availability and physical properties over stereolithography processes, it does have some shortfalls, there is a lack of resolution and a heated envelope is commonly used which can cause problems when incorporating multiple processes into an elevated temperature enclosure.

Material extrusion also known as Fused Deposition Modeling (FDM) is the process this paper will focus on when describing the AM technology for multi-functional parts. Material extrusion is where a thermoplastic is fed through a heated nozzle normally in a roll of filament or (less common) pellet form. In this case a roll of filament is used which is pushed through the system by a series of motors either a stepper motor or a DC motor with an encoder and then it is introduced into a heated liquefier that is attached to a nozzle. After it has been liquefied (above glass transition temperature), it is extruded through a measured nozzle and dispensed on to a platform for building. Figure 1.3 shows a diagram describing the Material extrusion process.

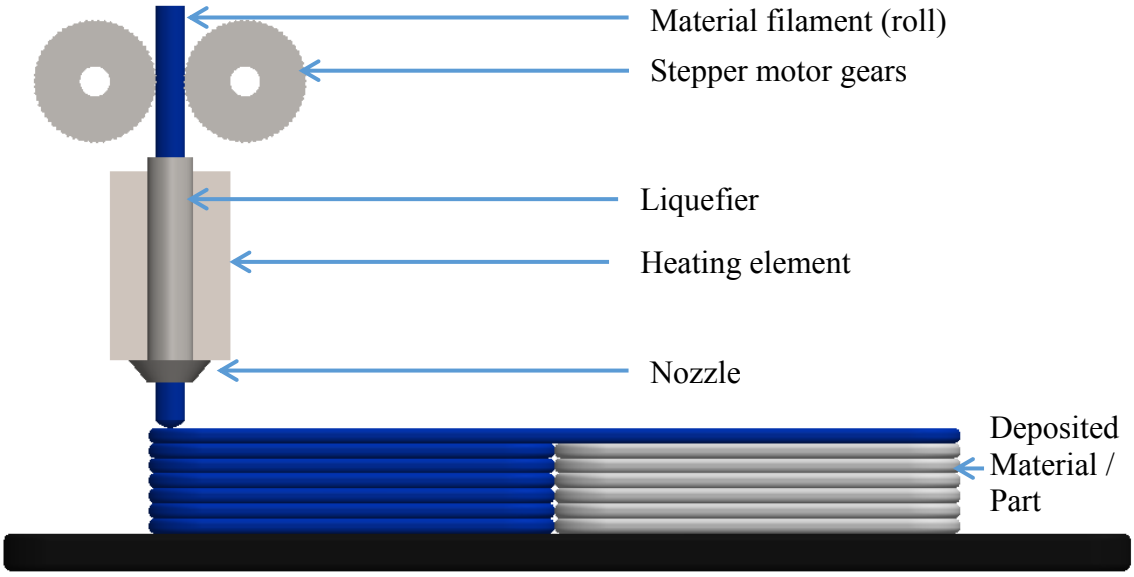


Figure 1.3 Material extrusion process

High-end AM machines that use material extrusion usually have two extruders one for the build material and one for a support material that can be removed post process as it helps with the build of certain geometries within a part. This extra material regularly increases the build time but also increases the success rate and yields superior parts with more intricate

designs. Typically, the post process used is a heated ultrasound with a solution that deteriorates away the support material while keeping the build material intact.

1.2 Motivation

One aspect both stereolithography and material extrusion multi-functional parts have in common is the incorporation of human interaction for the inclusion of components. Since the interest of incorporating foreign objects into AM parts started there has been a push for showcasing its viability, Figure 1.4 Shemelya *et al.* (2015) is an example of showing that there is a benefit for embedding electronics into an AM part. Seeing the benefits of multi-functional AM parts lays the path for creating processes that allow such intricate parts to be done in a simpler, faster, and more accurate manner.

Figure 1.4 Showcasing different antennas (Shemelya, 2015)

The process in which AM parts are created is straightforward starting with a Computer Aided Design (CAD) and being processed by a software package that is normally provided by the printer and finally loading the file into the machine and running it to build a part. This process is not built to include multiple technologies and walk arounds have been created for previous multi-functional parts. The development of a multi-functional and multi technology machine with a corresponding software package is being developed at the W.M. Keck Center for 3D innovation located at the University of Texas at El Paso (UTEP). The research presented in this paper will follow the course of creating and developing the basic process of generating multi-functional g-code. G-code is a Numeric Control programming language that is widely used

in Computer Numerical Control (CNC) machines to describe physical patterns; this in turn is also the way that most AM machines control their motors and in full their process. Although there are machines like Stratasys (Eden Prairie, MN) that use other forms to control and save their print files g-code is very popular and widely supported in the open source community. Therefore it does not matter in which Computer Aided Design (CAD) program the part is designed or in what software it is processed as it will end up as basic G commands, their might be formatting or encryption that differ from machine to machine but nothing that is extremely different. The software package and process being developed is also conjunctly being developed with hardware that uses basic g-code allowing it to be adapted to a greater number of machines.

1.3 Thesis Objective

The objectives of this thesis are:

1. Identify a process for creating 3D printed multi-functional parts
2. Create necessary software to achieve 3D printed multi-functional parts
3. Build on creating an interface for 3D printed electronics
4. Discuss and elaborate on the next step and where the technology is heading

1.3 Thesis Outline

There are four chapters dividing the paper. The second chapters will expand on AM and its capabilities, it will give a better insight on why the technology was chosen and why it is important to pursue such capabilities, including the software capabilities and where they are. The third chapter will go into detail on what is being done at the Keck Center to create a process for 3D printing multi-functional devices. Lastly, the fourth chapter will discuss the results and future work that will come.

Chapter 2: Literature Review

2.1 Introduction

Commonly referred to as 3D printing, Additive Manufacturing (AM) is a manufacturing process that builds parts in a layer by layer fashion. By having access to each layer, the complexity of the part being built does not affect the build process. Manufacturing AM parts regardless of the type of machine or material keep the same process steps. The main steps consist of creating a design on a computer using CAD software. Next, the designed is exported and processed through slicing software most AM machines provide one, open source software is also available and many use it. The created print file is commonly written in g-Code, which can then be loaded into the AM machine. Once it is loaded, the machine can begin to build the part, finally when the part is done printing there is a post process that is dependent on the technology and type of material that is being used.

2.2 Material Extrusion

One of the most common forms of additive manufacturing is that of material extrusion commonly known as Fused Deposition Modeling (FDM, trademarked by Stratasys Inc.). Although there are many forms of AM this paper focuses on the material extrusion method for which a strong ongoing effort for multi-functionality has been shown. Although, most of the reported electronics embedded in AM parts started with the use of photo curable resin technology paired with a conductive ink, parts described later on show a great promise using material extrusion techniques. There has been extensive research done in materials for additive manufacturing, much of the data collected is shown in Wholers Report (2013) the general consensus has been that the physical characteristic of thermoplastics tend to surpass those of photo curable polymers. The findings are also supported by Troger *et al.(2008) were* the durability of UV curable resins used in SL technologies is also studied.

The material extrusion process though simple in idea has taken many forms and has become an important aspect of printing to perfect (ASTM Standard F2792-12a. 2012). Most if

not all the parameters are based on the material that is being printed. The printing process starts with a CAD model on a computer, and then it is exported into a format that is accepted by the slicer like STL or AMF (more on file types later). The slicer is the software that converts the model into the file format that the printer can understand. The slicer slices the part into layers that will be printed individually one on top of the other. It is at this stage in the process that the user has the option to change the available parameters for that particular machine. Next, the slicer output file is loaded into the machine and the build process can begin. Lastly, when the process is done there is a post process involved depending on the machine it could be a simple removal of the part from the platform or if a support material was used an ultrasonic thermal bath in a solution is needed to remove all the support material.

A material extrusion based system works by introducing a thermoplastic into a liquefier using a stepper motor. Once the filament is inside the liquefier, heating elements raise the temperature of the filament to its glass transition temperature (TG) that allows the thermoplastic to become malleable and be pushed out through the nozzle of the extruder by the stepper motors that keep feeding the liquefiers. As the material is extruded through the nozzle it is deposited in a heated bed that bonds to it. An outline of the layer is created followed by an infill commonly known as rasters the density of these rasters are dependent on the parameter selected in the slicer and it is normally presented as a percentage. Once the layer is done the head stops extruding, moves to the next level, and begins on the next layer. The process depends on the molten rasters to attach to each other when they are still warm. With a heated bed and a heated envelope, the possibility of stresses and warpage decrease (Wang *et al.* 2007). Temperature control also helps with the adhesion from layer to layer as the layers stay warmer and bond better at higher temperatures. Figure 2.1 demonstrates what a common pattern and layout of a layer looks like.

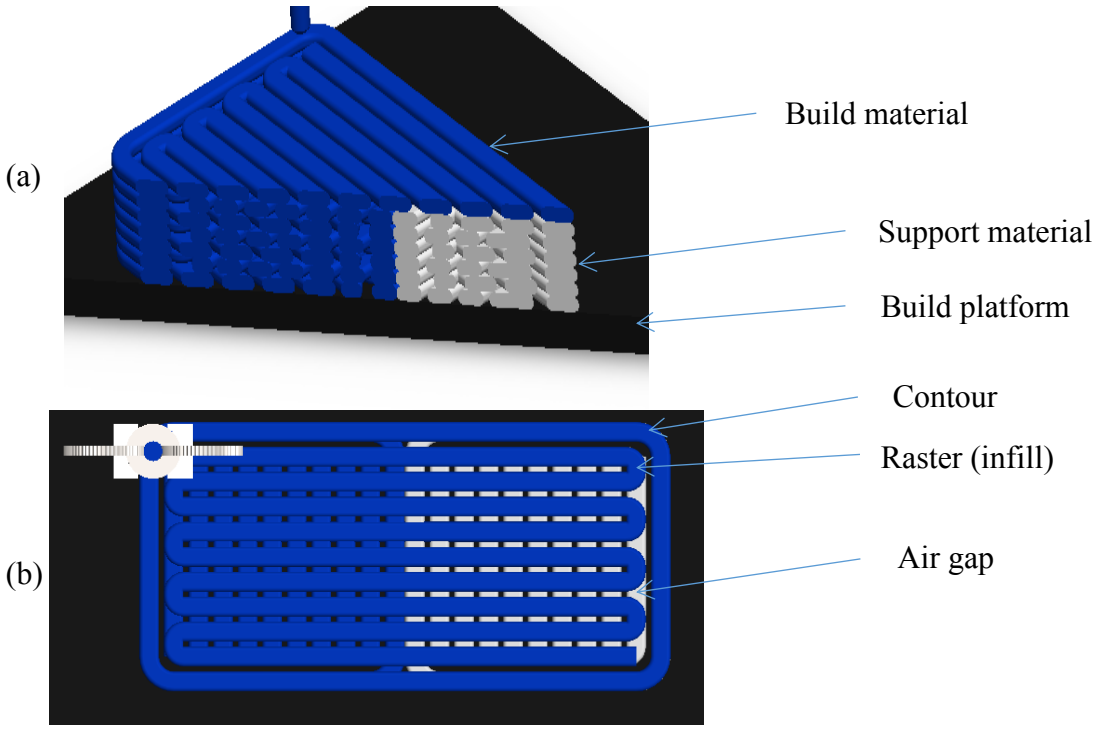


Figure 2.1 (a) Displays the cross section of a printed pattern including support material (b) shows a top view of the same part and the pattern of the dispense material

Every layer consists of a contour that becomes the shell of the part being built, inside the shell the raster is deposited as shown in Figure 2.1(b). The air gap is what defines the density of the part as a whole if the air gap is larger, then the part is less dense, also note the checkered appearance from the top view. Each subsequent raster is normally deposited at a different angle from its previous layer commonly 0/90, 30/-60, or 45/45 degrees, increasing the Ultimate Tensile Strength (UTS). Research by Hossain *et al.* (2013) shows that modifying raster angle, contour width, number of contours, raster width, and raster to raster air gap increased UTS for FDM parts.

2.3 Printed Electronics

End user parts fabricated through AM means have constantly been targeted with early reports in 1998 about the viability of the technology by Kruth *et al.* (1998). The same year Weiss

et al. (1998) started to experiment with embedding components and later Cham *et al.* (1999) and Katarina *et al.* (2001) described the same interest in embedding components for 3D printed electronics. In 2006 Lopes *et al.* was successful in combining the SL process and dispensing ink to create 3D printed electronic parts, with this advancement more complex electronics started to be produced at UTEP.

The University of Texas at El Paso has been one of the first research institutes with great success in building AM electronics using SL and conductive inks approach and now moving into thermal plastics. Starting with Medina *et al.* (2005) demonstrating the basic integration of ink circuitry in a stereolithography part. Later it was advanced in Navarette *et al.* (2007) with a wireless motion sensor and integrated GPS SL part depicted in Figure 2.2. In 2008 DeNava (2008) worked on a magnetometer application that showcased the freedom of sensor placement, and it was once again reworked in 2011 by Richard Olivas (2011) who also worked on other applications like a geometrically custom head insert with printed electronics.

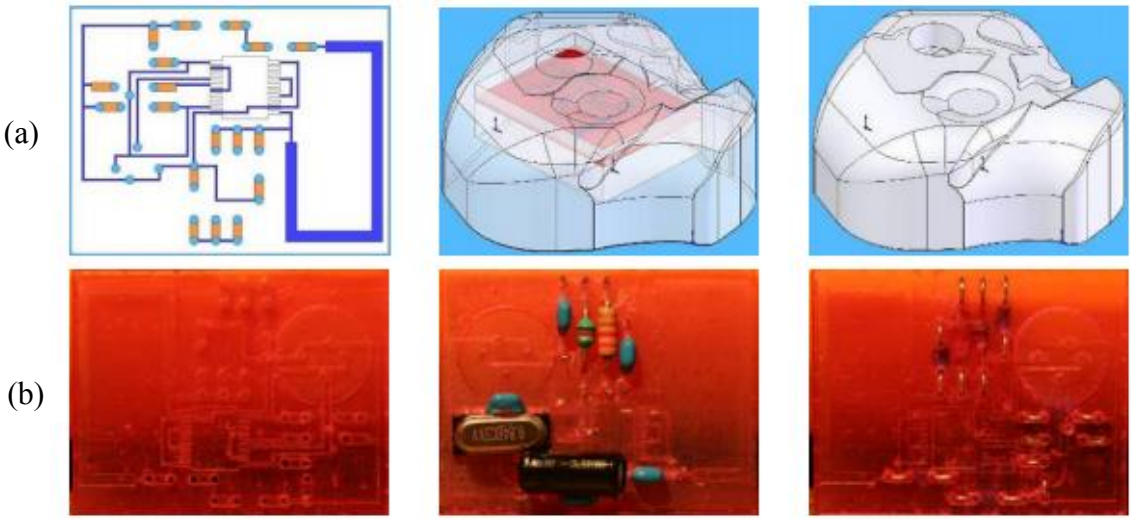


Figure 2.2 (a) Shows the CAD drawing of a sensor in SL while (b) showcases the actual part being built with electronics

In more detail, DeNava (2008) worked on a magnetometer application that showcased the freedom of sensor placement, with hall effect sensors throughout the perimeter of the part, the main circuitry in a different plane, and the battery on the opposite plane of the main circuitry.

This helped promote the usability and demonstrate the capabilities of 3D printed electronics. The magnetometer with improvements by Olivas (2011) in which the general shape was changed and improved positioning of the sensors indicates how 3D printed parts can be iterated. The magnetometer has an array of hall effect sensors through its perimeter with Light Emitting Diodes (LEDs) that give visual feedback on the direction of the magnetic field, with another set of LEDS that display the magnitude of the magnetic field. Figure 2.3 portrays the evolution of the magnetometer from its first iteration in 2008 to its last in 2011.

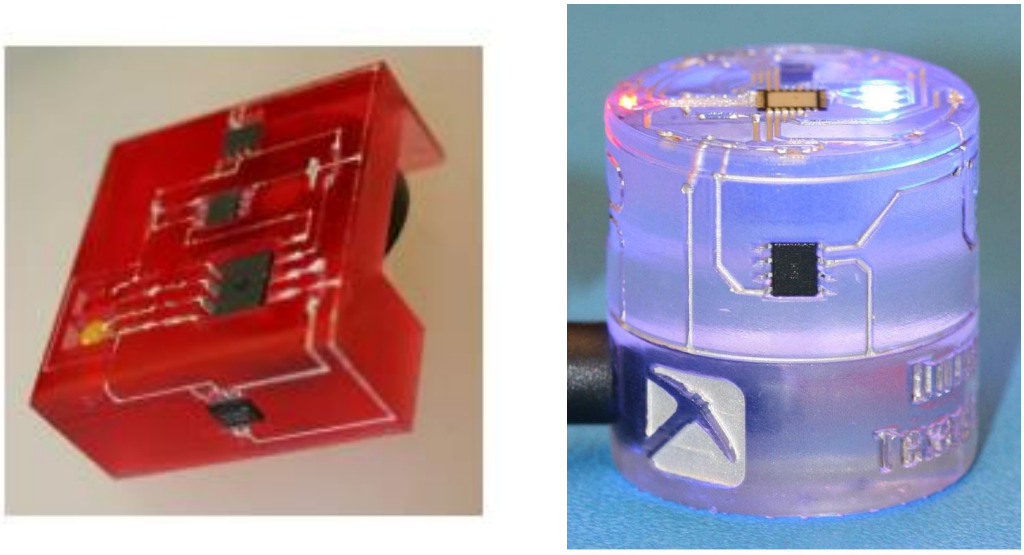


Figure 2.3 (a) First iteration of the Magnetometer (b) last iteration of the magnetometer

One of the most considerable advancements in AM electronics using SL and conductive inks came with the creation of a CubeSat module. The CubeSat module was created at UTEP in partnership with New Mexico's Cosmiac and the University of New Mexico. The module has a fully embedded circuit within itself that supported communication to the rest of the device and also had an integrated gyroscope to capture orientation and a resistive pattern in the underside to help characterize the effects of space on 3D printed circuits. Figure 2.4 shows the 3d printed CubeSat module that was developed at UTEP. The module was sent into low orbit space in the Trailblazer U2 CubeSat in November of 2013.

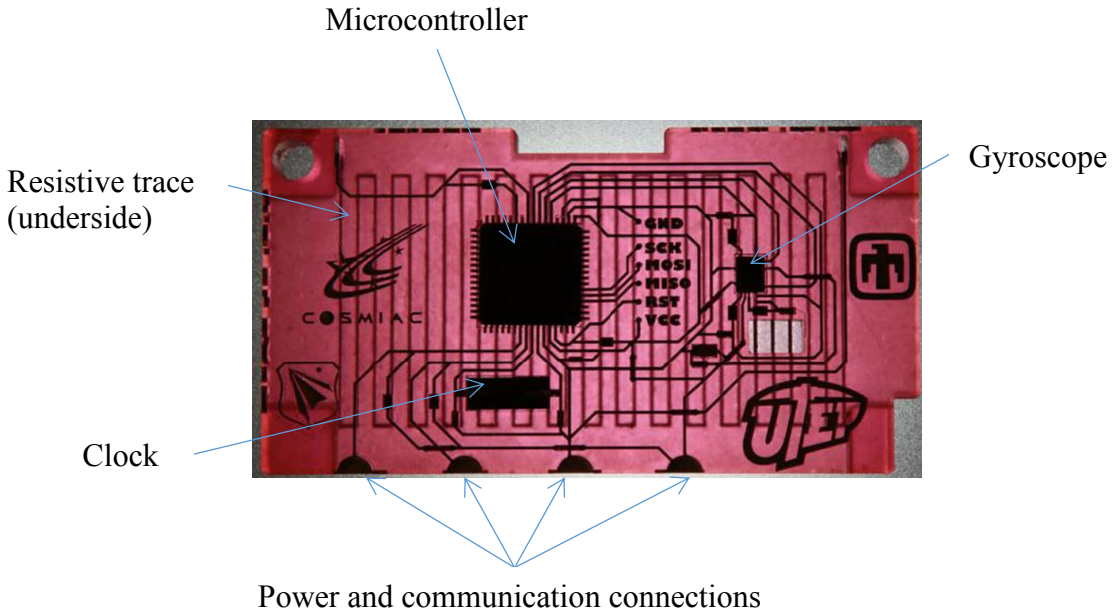
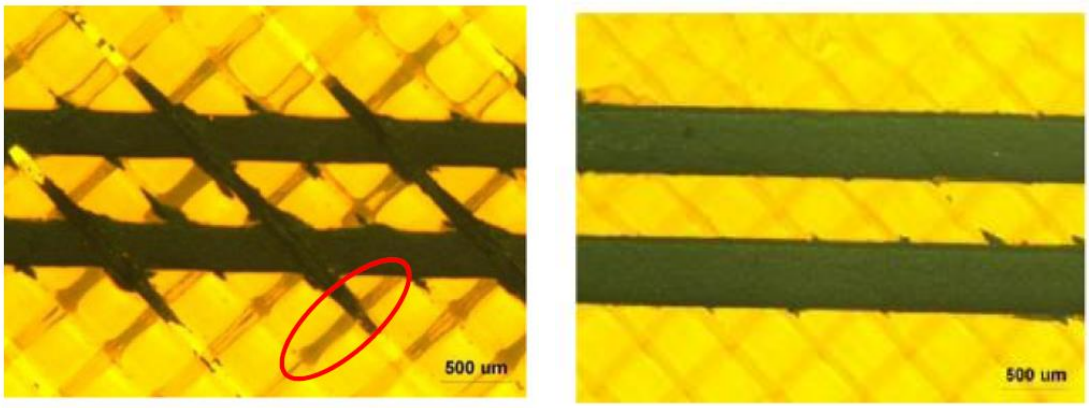


Figure 2.4 CubeSat module developed by UTEP with major components labeled

As the popularity of 3D printed electronics increased, it started to pique the interest of more people and other methods started to be explored. In an effort at Keck Center and with the focused on thermoplastics for printed electronics, a new way of electrical interconnects was developed. Although inks work and have proven an effective way of creating AM electronics they do inhibit certain applications as described in Aguilera *et al.* (2013). One of the first attempts to create 3D printed electronics using material extrusion was migrating a project from SL, meaning the only thing changing is the substrate. Espalin *et al.* (2014) explains that using conductive inks in material extrusion can be difficult because of the porousness of the part being built, the ink dispense in its liquid form can run along the rasters and short. There are ways to mitigate these issues like micromachining channels for better resolution and finish while also maintaining a minimum distance between each trace to reduce shorting. Looking at Figure 2.5 the issues encounter are shown before and after the process changes that help mitigate the issues.



(a) (b)

Figure 2.5 (a) Conductive inks shorting in paths that were machines into the plastic. (b) after process parameters are modified for conductive inks in plastics

With these problems solved the need to expand the usability (high power density) of 3D printing led to the integration of direct wire embedding for higher power application. A comparison in electronic interconnects was performed in Espalin *et al.* (2014) shown in Figure 2.6, the performance of Printed Circuit Boards (PCB), conductive inks, and embedded wire are compared, it can be deduced that embedding solid copper wire has better characteristics than both inks and PCB, which is to be expected.

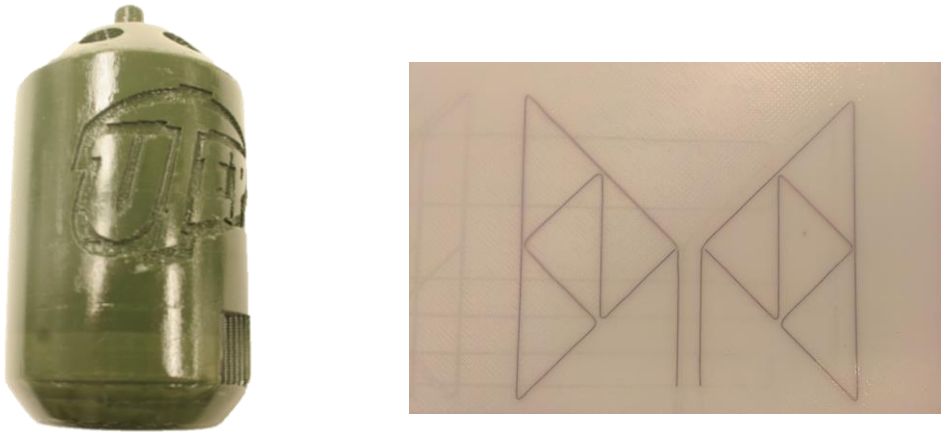
Case	Geometry	Resistivity (Ω m)	Resistance (Ω)
1 oz copper PCB with 4 mil width	37 μm thick, 100 μm wide, conductor 10 cm long	1.7×10^{-8}	0.45
Dupont Ink CB028 Silver	25 μm thick, 100 μm wide, conductor 10 cm long	11.8×10^{-8}	4.73
Dupont Ink CB500 Copper	25 μm thick, 100 μm wide, conductor 10 cm long	50.7×10^{-8}	20.27
Extruded solder	25 μm thick, 100 μm wide, conductor 10 cm long	7.2×10^{-8}	2.86
40 gauge wire 10 cm long	80 μm diameter, conductor 10 cm long	1.7×10^{-8}	0.33
32 gauge wire 10 cm long	200 μm diameter, conductor 10 cm long	1.7×10^{-8}	0.05
28 gauge wire 10 cm long	320 μm diameter, conductor 10 cm long	1.7×10^{-8}	0.02

Figure 2.6 Table showing the comparison of resistivity and resistance of a PCB trace, conductive ink, and solid copper wire.

Embedding solid copper wire can be done multiple ways, the preferred method and used at the University of Texas at El Paso is a direct heating method. Similar to that of extruding

material a block of metal is heated up with some electric heating elements. Through the heated area the copper wire to be embedded is passed through, as the wire passes through it heats up then as it is introduced into the material it melts into the substrate and quickly cools allowing the dispersed material to cover it and hold it in place. The wire can be assisted by a motor but it uses the same tension of the initial embedded wire to help feed more wire through the heated head.

Being able to embed and route wire into an AM part allows the part to become more structurally sound, Espalin *et al.* (2014), and they can act as electronic interconnects. Examples in Shemelya *et al.* (2015) of the functionality created by integrating wire and mesh into AM parts, antennas can be created that are spatially accurate by modifying the density of the substrates as needed (MacDonald *et al.,* 2016). Figure 2.7 demonstrates another application in which the use of conductive inks would not be possible because of its low conductivity, and capacity of carrying amperage.



(a) (b)

Figure 2.7 (a) UTEP's 3D printed motor version 2 using wire embedding technology. (b) Embedded wire pattern

Other groups and companies have been pursuing the integration of additive manufacturing with electronics, Voxel8 (Somerville,MA) a startup from Harvard has integrated

conductive ink dispensing with material extrusion to provide a single AM machine that is capable of producing 3D printed electronics. Figure 2.8 demonstrates a quadcopter build that the voxel8 machine can produce, it connects the control PCB that is placed in the center of the build part to the motors that are placed on the outside, then it continues and covers the deposited traces with more polymer to complete the build.

(a) (b)

Figure 2.8 Voxel8 machine printing a quadcopter with electronic traces. (a) electronics inside the structure, (b) finished quadcopter from Voxel8 (pictures from www.voxel8.com)

2.4 G-Code

Most AM machines including the multi-technology machines being developed in the Keck Center for 3D innovation are based on g-code. g-code is better known in the Computer Numerical Control or CNC world, it is used to programmatically tell the machine where to move and what to do. Although it is a very simple programming language it has evolved through the years to accommodate other functions. G-code has gone through three main iterations the standard is owned by UCAMCO (www.ucamco.com, 2016) and they have the full documentation. It started with RS274-D, which is the standard Gerber file, and it was the first file format developed. It has since been left and it is not supported any longer by UCAMCO, the standard moved to RS274-X or more commonly known as the extended Gerber file format. The move was done to add support to more commands both standard g commands and new "extended" commands. Recently the move to RS274-X2 or an upgrade to RS274-X has started to

be implemented and is considered the latest Gerber format. Although the Gerber format has been updated the basic formatting has not changed other than adding extra functionality and commands. Since the actual formatting has not been changed many companies creating motion control drivers identify their interpreters as RS274 meaning most basic g-code commands can be used and they normally provide a comprehensive list of commands supported by their drives. In general g-code is a very simple and linear machine programming language, Figure 2.9 demonstrates what a basic g-code file looks like. In its basic form the commands are move commands in the Cartesian plane with extra axis described by E commands for coordinates, which in 3D printing is usually to control a stepper motor for the material feed. Looking at the standard provided by UCAMCO the complete list of commands can be found, although it is better to look at the drivers that the system is using because that is what the system actually supports.

```
29  G1 F1200 X171.443 Y127.736 E0.76991
30  G1 X171.443 Y147.386 E1.10499
31  G1 X126.293 Y147.386 E1.87490
32  G1 X126.293 Y127.736 E2.20998
33  G0 F9000 X126.643 Y128.086
34  G1 F1200 X171.093 Y128.086 E2.96796
35  G1 X171.093 Y147.036 E3.29110
36  G1 X126.643 Y147.036 E4.04907
37  G1 X126.643 Y128.086 E4.37221
38  G0 F9000 X126.993 Y128.436
39  G1 F1200 X170.743 Y128.436 E5.11825
40  G1 X170.743 Y146.686 E5.42946
41  G1 X126.993 Y146.686 E6.17550
42  G1 X126.993 Y128.436 E6.48670
43  G0 F9000 X127.343 Y128.786
44  G1 F1200 X170.393 Y128.786 E7.22080
45  G1 X170.393 Y146.336 E7.52007
46  G1 X127.343 Y146.336 E8.25417
```

Figure 2.9 Example of what g-code looks like for a material extrusion 3D printer

2.5 3D Printing Software

Since the beginning of AM technology there has been development in both hardware and software for AM machines. The software is as important as the hardware, it allows us to automate and easily build parts. As technologies advance and become more complicated the process (software) also evolves and becomes more complicated. AM is a type of digital manufacturing and it shares similar processes as that of traditional manufacturing. It all starts in a CAD drawing on a computer, then moves to Computer Aided Manufacturing (CAM), which helps transfer the CAD into machine code. This process is observed in plastic injection molding and in machining, many times a combination of those technologies are needed to create a single product. Where the differences start to become apparent is in the CAM process, the files need to be prepared specifically to a particular AM machine. The goal is to go from a CAD on a computer to a file in g-code that controls the machine and directs it accordingly to manufacture a part. This process is achieved by exporting the CAD into a file format that is accepted by the intermediate software normally referred to as a "slicer". One of the most common file formats is the standard tessellation language or STL for short, it is a basic file that only represents the surface of the object. It does so with an unordered list of coordinates that represent triangle vertices, because each triangle is represented separately each vertex that is shared (normally all of them) are repeated. STL has been serving this information transfer for most of the AM life (Kumar *et al.,* 1997), as the technology has left the simplicity of single material extrusion and is moving into multi-material and multi-functionality STL files are becoming too simple. Hiller *et al.* (2009) describes what a file format must have in order to become a new standard formatting for AM. Hiller request the file format be technologically independent, simple, scalable, and future compatible, he seems to propose a new file format called Additive Manufacturing File or AMF. The AMF format is based on Extensible Markup Language (XML) and it is capable of holding all of the information that an STL file already holds while allowing for more information to be transfer. Multiple materials can be defined and is one of the more basic and useful extra information used as of now. With the ability to add extra tags (XML capability) to the file multi-

functionality can be stored. Tags representing circuits, machining, and other functionality can be added although at this moment there is no standard for adding multi-functionality. AMF does have a standard for describing the physical part with a hierarchy of top level tags describing the parts (ISO/ASTM 52915:2013). Constellations describe the objects in the file and give it ID's or ways to define them. There is an object tag for each part described in the constellation holding the vertex and physical coordinates. Palette defines the materials to use in each object or part and lastly the Print tag gives extra information on how to print or how many to print. An example AMF file describing a basic cube is shown in Figure 2.10, the file was created in SolidWorks (SW)2015. If information is not present like the palette tag or print then it omits them and assumes the basics of printing in one material Hiller *et al.* (2009).

Figure 2.10 AMF file example of a cube created with SolidWorks 2015

Software for slicing, which is what accepts the exported STL or AMF file from the CAD is normally provided by the machine manufacturer, but there are open source programs that work for g-code base machines which tend to be more open-source friendly. These slicing software's can have a fair amount of variables to modify parameters. As seen in Hossain *et al.* (2013) using

a Stratasy's machine with the provided Insight slicing software changing raster gaps, angles, and contours one can change properties of the printed parts. These slicing software's are still primitive for multi-functional parts, there is a strong interest on software for multi-functional machines yet without the hardware support it becomes difficult to work on. Autodesk (www.Autodesk.com) a company based out of California has been creating CAD programs like Fusion 360 and has expressed interest in creating multi-functional CAD's for AM technologies. Autodesk paired initially with Voxel8 to develop Project Wire, which is interactive software to add electronics and interconnects to CAD models that can then be exported into Euclid their own slicer that produces a file that can be used by the machine. Figure 2.11 depicts Project Wire Graphical User Interface (GUI) that helps facilitate the integration of electronics and interconnects.

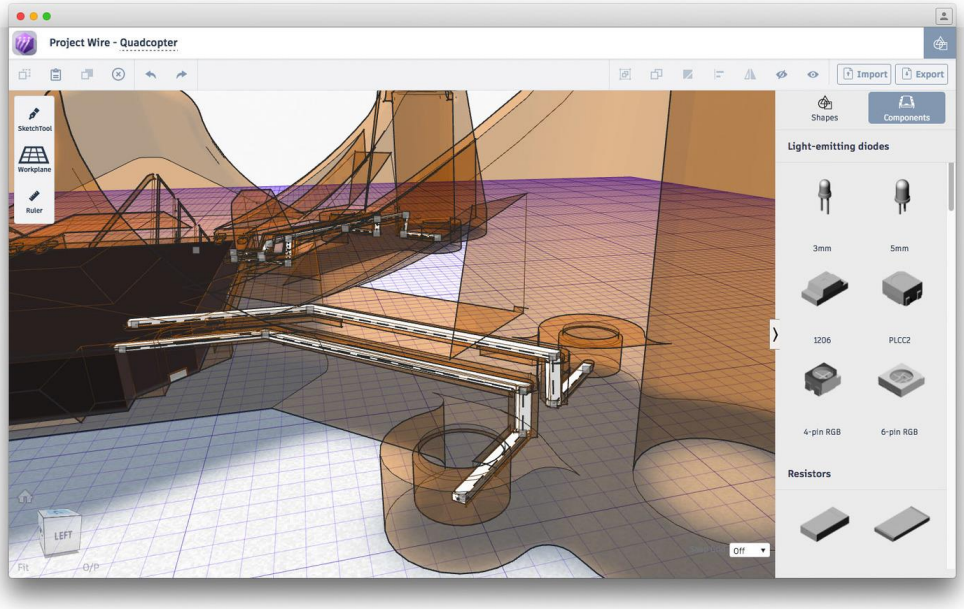


Figure 2.11 Project Wire GUI for integration of electronics and interconnects (courtesy of wwwVoxel8.com)

Wasserfall (2015) has been strongly researching software to implement 3D electronics in AM parts. The efforts seem to be closer related to those with Voxel8 as the technology used material extrusion and conductive inks are similar if not identical. Wasserfall seems to run into

the same issues of having a lack of support and is experimenting with computer vision to align components he has also done extensive researched showed in Figure 2.12 with the use of 45 degree slopes to create vertical interconnects. A student under Wasserfall supervision Ahlers (2015) worked on implementing a software system for 3D printed electronics. In which he came into the conclusion and demonstrated that integrating electronics into AM parts was best done at the slicing software and used Slic3r as his starting point. Although he was successful at implementing components into the slicer, routing and interconnects was a limitation as it had to be done by hand.

(a)

(b)

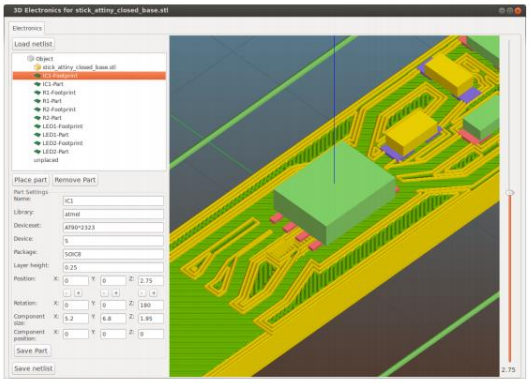


Figure 2.12 (a) Wasserfall work for 3D printed electronics including vertical interconnnects, (b) Ahlers work using Slic3r to implement components into AM parts

2.6 Conclusion

Through the recent explosion of interest in AM there has been many advances and new interest being deveopled. One interest is to use AM for multi-functionality, mainly creating 3D printed parts with electronics embedded into them. 3D printed electronics have come in many different forms from conductive inks with SL to embedded copper wire in material extrusion, all

with the same goal of giving AM parts more functionality. It is clear that there is a lack of work being done in supporting multi-functionality from a software point of view, with the interest of expanding on software for AM multi-functionality this thesis will explore what is needed to go from a CAD to a 3D printed multi-functional part.

Chapter 3: Unification of G-code for Multi-Functionality

3.1 Introduction

In the following chapter, a detailed description of the work being done at the Keck Center for software in relation multi-functionality parts will be explained. As UTEP has developed manufacturing methods and useful applications for multi-functional parts, interest of creating a single automated multi-functional machine have been expressed. With the awarded grants from America Makes, a proposed multi-functional machine is being built at the Keck Center. The capabilities include multi-material, wire embedding, subtractive manufacturing (CNC), foil embedding, and robotic component placement. As part of the requirements, a software package has to be developed.

This thesis encompasses the ongoing efforts of developing software for multi-functional parts, due to being a large project there are multiple people working towards the same goal. This thesis goes into detail on what the author has developed, the status of the work, the direction of the project, what has been done before, what is being done by others, and what will be done in the future. One of the most promising aspects of multi-functional parts is the capability to embed electronic components, and UTEP has pushed 3D Printing as a viable manufacturing process therefore the first demonstration in software will focus in creating and processing 3D printed electronics.

3.2 Software Requirements and Constraints

Requirements

- Unify Technologies
    - Material Extrusion
        - Multi-material
        - Slicer
        - Pauses
    - Embedded Electronics

        - Wire routing
        - Component placement
        - Software package
            - Integration of ME and EE designs
    - CNC/Machining
        - Tools
        - Paths
    - Foil Embedding
        - Process
        - Paths
    - Machine Specific
        - Lulzbot
        - Multi3D (low-cost)
        - BAAM (Big Area Additive Manufacturing)

Constraints

- G-code output
- File Format (AMF,STL)
- Slicer (open source)
- Use existing software
    - SolidWorks
    - Eagle
- New hardware

3.3 Thesis Goals

The project's software requirements and constraints have been defined as a whole, a team of students and engineers at the Keck Center are working towards achieving these. The work presented in this thesis follows the beginnings of the project and the foundation laid for future

development. The approach was a simple systematic approach to build a foundation that can be expanded and worked on in parallel for quicker development times. The main goals focused on for the software package are dependent on the previously listed requirements and constraints and are listed below.

Requirements:

- Software Package
    - Reads multi-functional file(s)
        - SolidWorks with electronics components
            - Component placement
            - routing
        - Export all relevant information
    - Create a single file output (g-code)
    - Minimal user interference

3.4 Early Developments and Decisions

A comparison of the process of creating a 3D printable file versus creating a 3D multi-functional file is demonstrated as a flow chart in Figure 3.1. SolidWorks has been chosen as the CAD software for this project because of its prominence in the industry, and as one of the resources received by the university. SolidWorks has the ability to export STL files, STL files are among the most popular file types when designing parts for AM. Recently SolidWorks has adopted the AMF file type (2015), implementing a direct file for AM shows the interest SolidWorks has on AM technologies. Although the AMF file created by SolidWorks supports multi- material there is a lack of support for multi-functionality.

With SolidWorks as the chosen program to design multi-functional parts there are certain pieces of information that need to be extracted from the CAD model to manufacture multi-functional parts. As described before the two main file types that SolidWorks uses to export dimensional information for AM is STL and AMF, both in this case hold the same information,

also the slicer program used in this project accepts both types. Therefore, at this early development stage either file format suffices, the next step is to identify the information needed to be extracted.

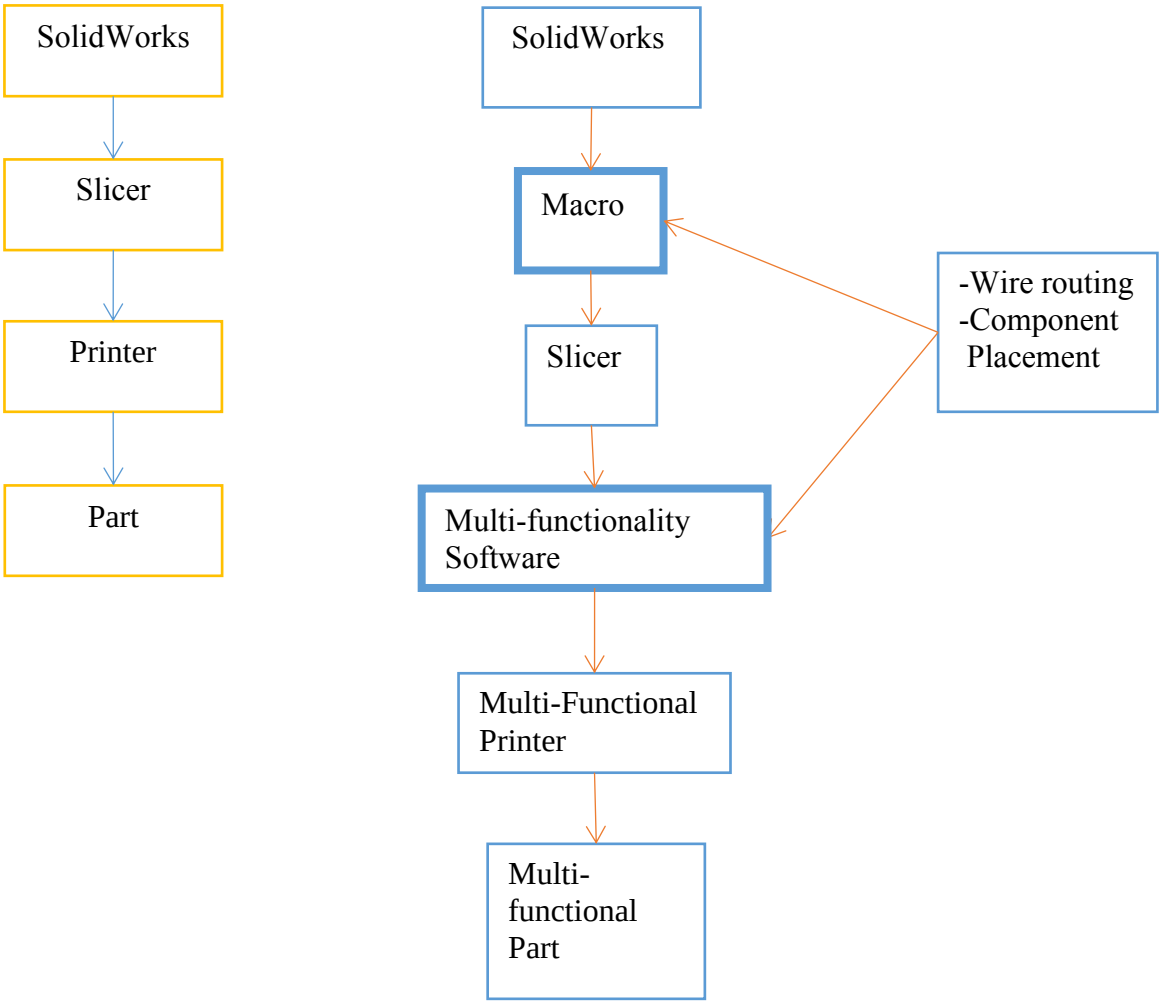


Figure 3.1 High level flow chart demonstrating steps to create a normal 3D printed part versus the steps needed to create a multi-functional 3D printed part

There are several key pieces of information that have to be extracted from the CAD model in SolidWorks, listed below.

- Circuit Path
- Height

- Component placement

Looking at the list it seems rather simple yet each function is not trivial to implement. There is a push to integrate EE CADs and ME CADs, keeping this in mind there are not many ways that a circuit can be represented in SW. From the three types of information that need to be extracted for 3D printed electronics emphasis is placed in the first two (circuit path and Height). Component placement is important but not vital; it will be discussed in a more detailed approach for integrating EE schematics later. SolidWorks has an Application Program Interface (API) that interfaces through Visual Basic for Applications (VBA), using VBA with the SW API macros can be created in a very similar way to macros written for Microsoft Word or Excel. First looking into circuit paths, a basic sketch with lines can represent "wires" or connections between components furthermore these sketches can be isolated and exported using the Drawing Exchange Format commonly known as DXF output. This creates a single file that holds information of the circuit path. With every circuit that is exported, it is paired with a height in which the layer can be found. Extracting the height from SW can be an intricate process, when a user enters into a normal 2D sketch it defaults to a basic x and y coordinate plane where z or in this case height is not defined. Therefore, height information cannot be extracted from a 2D sketch using a macro as it defaults to zero. SW allows the creation of 3D sketches and this mode stays true to 3D with x, y and z values, it is important to note that SW comes with a standard origin and three pre-defined planes top, front, side planes. As oppose to the normal convention of x and y being the horizontal axis and z being the vertical or "height", SW uses z and x as the horizontal axis and y as the vertical axis. Figure 3.2 is a screen shot of SW depicting the origin from a Top point of view.

When using 3D sketches the user has the option to choose a plane and reference that plane to a 3D sketch, making it easier to design on a single plane with a 3D sketch. Once the circuit is created using the 3D sketch the sketch is renamed, a generic name of "circuit#" is given were "#" is replaced by a number that refers to the amount of circuits being created, for example

the first circuit would be “circuit1”, the second circuit would be “circuit2” and so on. Figure 3.3 illustrates what a design tree for a 3D printed part might look like.

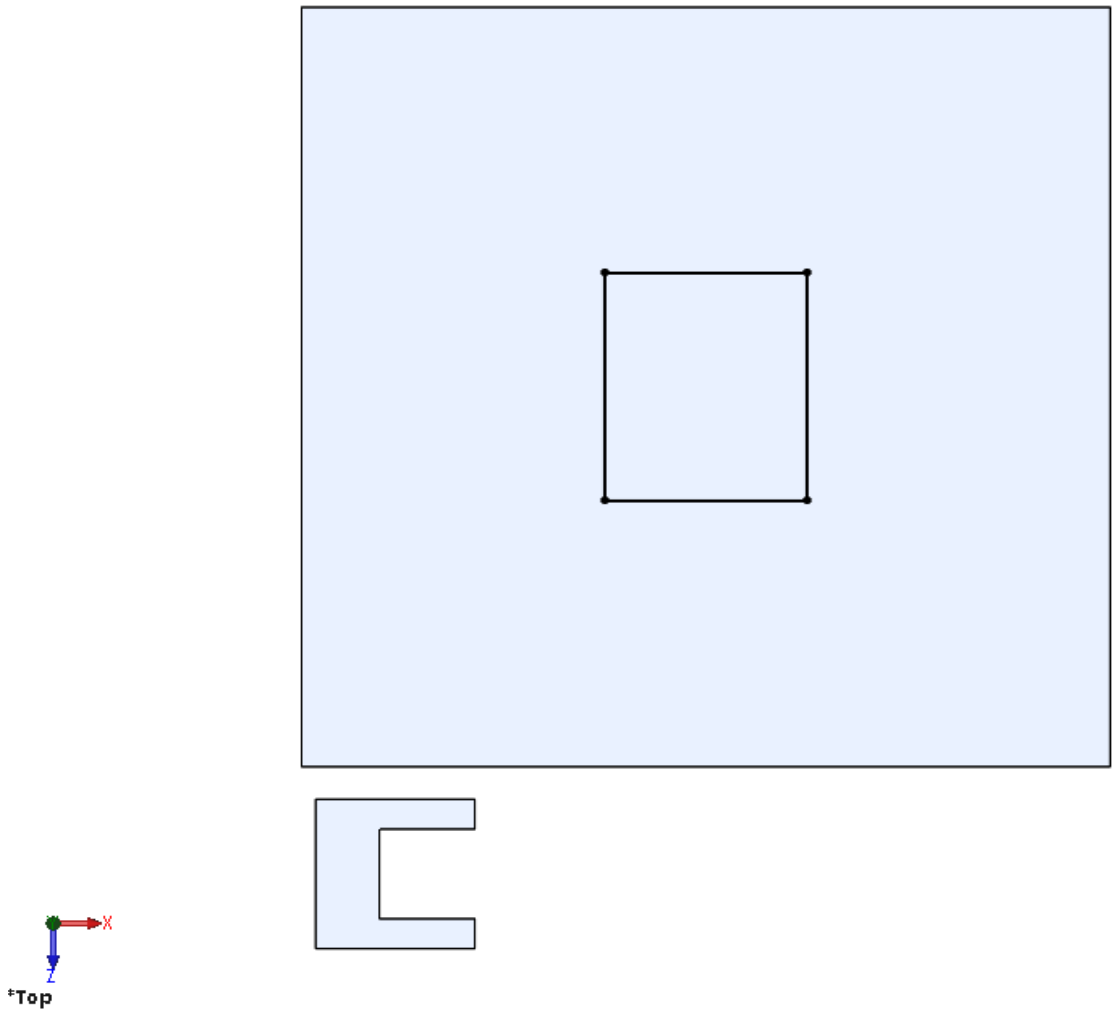


Figure 3.2 SolidWorks origin from the top view. XYZ depicted in the bottom left corner

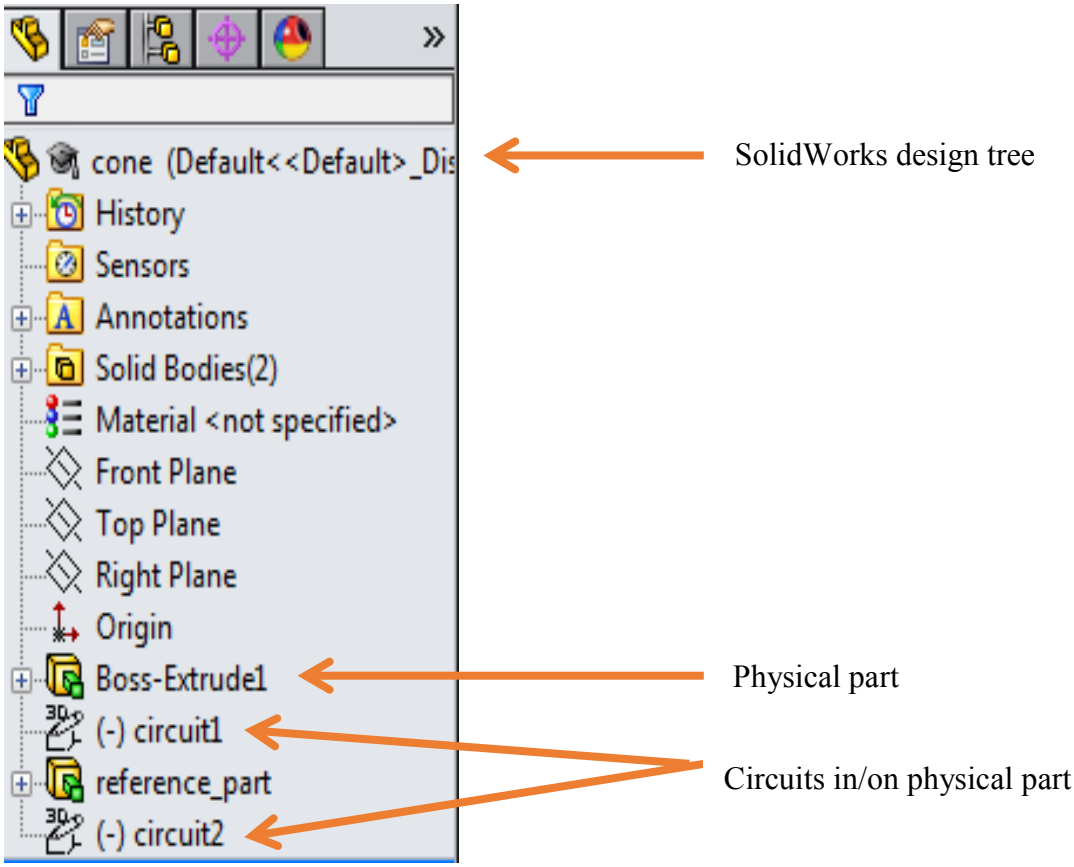
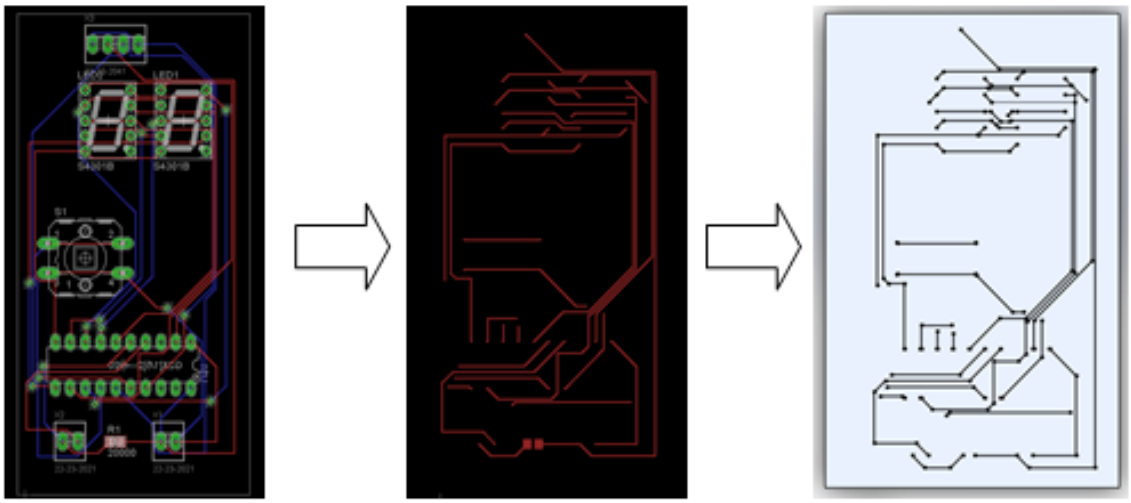


Figure 3.3 SolidWorks Design tree with showing how circuits are defined

Another key aspect about creating circuits is the ability to import DXF files into SW sketches, when a DXF file is imported into a SW sketch it is converted into lines. Programs like EAGLE, which help design PCB's output files very similar to g-code and normally different layers can be exported as DXF files. This allows for the creation of electrically sound circuits that can be then imported into SW. Figure 3.4 is an example of a circuit designed in EAGLE, exported out as a DXF file and then imported into SW where it can be placed into the part that is being designed.



EAGLE PCB design | Single layer to be exported | Single layer imported into SolidWorks model

Figure 3.4 EAGLE CAD exported as a DXF file and imported into a SolidWorks model.

3.5 SolidWorks Macros

The process of defining and creating a circuit in SolidWorks is well established, details in the macro can be better understood. The macro is a standard Visual Basic program, it starts with basic definitions of variables and objects. The program first grabs the location of the file that is currently open and creates paths describing the files that will be created using the macro. There continues to be more definitions for the type of views that will be exported through the DXF format. The top view is selected because it is the view that is perpendicular to the part's height. Once that is done all of the bodies are hidden, this is done to so that when the circuit is exported only the visible sketches are exported, a reference is needed for alignment purposes described shortly. The reference sketch is made visible, at this point the program is ready to export. A folder is created to store all of the files that are about to be created, the name is based on the part that is opened with the addition of "_project" for example if the part is named "part1" the folder name is "part1_project" and the folder is created at the same level of the part. If a folder already exist it assumes that it is a previous version and it will use it and overwrite the files inside. Once

the folder is created the text file “info.txt” is opened, this is the file that can be used to add any type of extra information, in this case it is used to store the heights of each circuit being exported. Following the creation of the file the macro goes and selects every “circuit#” sketch and hides it. This is needed for the same reason that the bodies are hidden, whatever is visible will be exported. At this stage only the sketch for the reference should be visible because the reference should come out with every circuit. The program goes into its main “do while “ loop. the loop will grab the first circuit sketch making it visible then extracting its Y coordinate which is the height of the circuit with respect to the part. The program then writes the first circuit with its corresponding height, then it exports it using the defined views which are set for the top view. This allows the user to not worry about aligning the view to the top, making it easier to use the macro. It saves the DXF file with the same name of the sketch, which allows easy matching in later steps. When it is done exporting, it goes and makes the current sketch invisible and then loops and looks for the next sketch, doing this until all the sketches have been exported. Once it reaches the last sketch it comes out of the loop and closes the text file. It then makes all the bodies visible and makes the reference sketch invisible. Figure 3.5 holds the folder structure and info file that is created from a sample 3D printed electronic part.

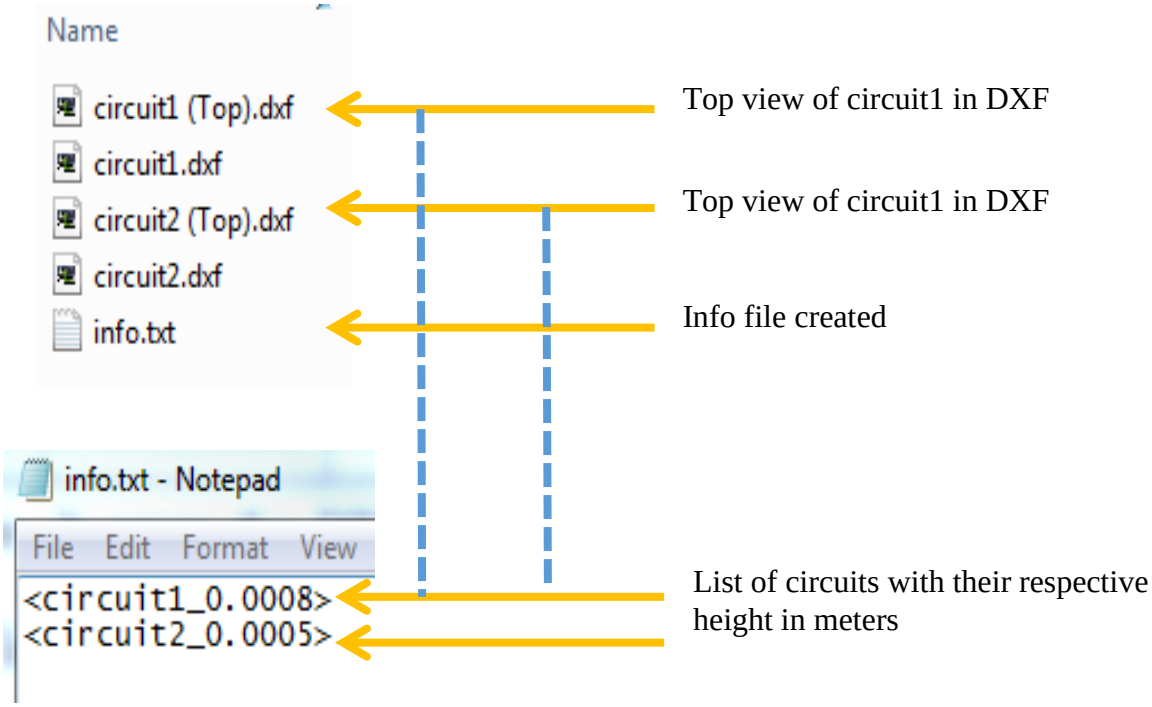


Figure 3.5 Folder created to store project files and contents of the info.txt file

A library of parts was created to test the macro, this is used as a qualitative test for the macro although measurements can be recorded for a more thorough comparison. The library contains 20 SolidWorks models that where built in a random way, half of them use the lines sketch property to create the circuit. While the other implements curves into the circuits, this is important to note as the difference between the way that SolidWorks exports lines and curves is different which is interesting to see how the g-code is created. Moreover, all the sketches in the parts are located on different layer heights.

Before testing starts another macro was developed to create a reference. A reference shape is created inside the CAD that is used by later processing algorithms, this reference part is automatically created and manually placed by the program and user respectively. Figure 3.6 shows the steps that are needed for the reference shape to be placed successfully.

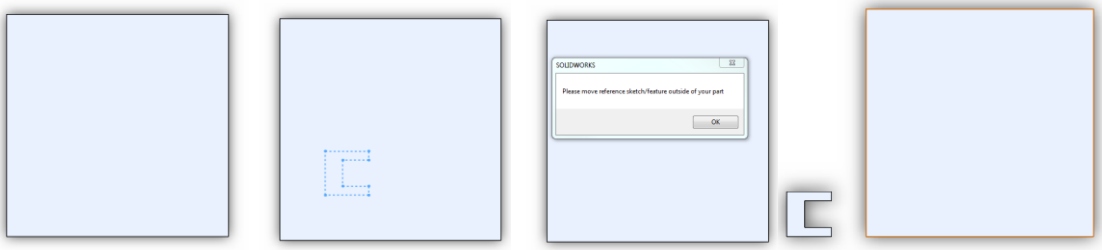


Figure 3.6 Process to create reference shape inside SolidWorks. (a) shows a basic model created in SW. (b) shows the macro creating the reference shape sketch. (c) after the macro extrudes the reference shape it lets the user know to move the sketch outside the model if they are interfering with each other. (d) shows the finished model with the reference outside the model.

The reference shape macro starts by selecting the top plane of the SW model, this is done because the top plane also serves as the platform or "zero layer" of the model. The reference shape should start at the lowest level so it doesn't not interfere with anything else. Once the macro selects the top plane a sketch is drawn in the shape of a blocky C, this shape was picked because of its simplicity and it ability to give us rotational information. After the sketch is drawn at the origin then it is extruded a small amount in the positive Y direction. A pop-up comes up that informs the user to edit the sketch and move it somewhere outside the model. Lastly, both the sketch and the feature created for the reference have the name "reference".

The reference shape is essemtial because of the unfortunate lack of support by CAD software. The extra information that is needed to create multi-functional parts has to be extracted and such extraction is not naturally available therefore much of the information that ties everything together is lost. One key piece of information is the orientation of the circuit, when

the model is exported as an STL or AMF it loses its orientation, when the circuit inside the part is exported it also loses the orientation, and when it is imported into the slicer the user can rotate and move the piece around the platform. At the end there are two sets of coordinates (model and circuits) that are similar at a layer but can be located anywhere in space or in this case the Cartesian coordinate system. Using a reference shape that stays consistently relative to the model and the circuit, a relationship can be formed from each identical reference shape and that same relationship applies to the actual model and circuit.

3.6 SolidWorks Macro Output and Testing

When both of the macros are run with the reference macro first then the export following, a folder holding all the information needed to create 3D printed electronics is created. Inside this folder the information includes DXF files containing the circuit paths with a reference shape for alignment. In addition, an information text file contains the name of the circuits and their corresponding heights. Although, SolidWorks allows to change units when designing parts the API defaults into meters, meaning all measurements have to be converted into millimeters.

Testing of such macros involved a more qualitative than quantitative approach. Although, capturing the macros accuracy in exporting is as important as knowing the macro runs across a variety of different models created in SolidWorks. Table 1 holds several criteria to evaluate the macros, a simple pass/fail to determine if the macros ran accordingly, and recording the height of the circuit in the CAD model and the exported file, lastly a visual inspection of the DXF files and SW circuit to make sure that the files hold both the reference shape and the circuit. Appendix B holds the visual inspection table.

Table 1 Testing Reference and Exporting macros

Model #	Reference Macro	DXF Export Macro	SolidWorks Height (mm)	Info File Height (mm)	Visual Inspection
Curves 1	Passed	Passed	13	13	Passed
Curves 2	Passed	Passed	13	13	Passed
Curves 3	Passed	Passed	30	30	Passed
Curves 4	Passed	Passed	25.5	25.5	Passed
Curves 5	Passed	Passed	5.5	5.5	Passed
Curves 6	Passed	Passed	40	40	Passed
Curves 7	Passed	Passed	19	19	Passed
Curves 8	Passed	Passed	7	7	Passed
Curves 9	Passed	Passed	17	17	Passed
Curves 10	Passed	Passed	7	7	Passed
Lines 1	Passed	Passed	20	20	Passed
Lines 2	Passed	Passed	14	14	Passed
Lines 3	Passed	Passed	9	9	Passed
Lines 4	Passed	Passed	9	9	Passed
Lines 5	Passed	Passed	5	5	Passed
Lines 6	Passed	Passed	30	30	Passed
Lines 7	Passed	Passed	3	3	Passed
Lines 8	Passed	Passed	13	13	Passed
Lines 9	Passed	Passed	60	60	Passed
Lines 10	Passed	Passed	34	34	Passed

The model number column defines what type of model it is for each category, “curves” is used to describe a model that implements curves in its circuits, while “lines” is used to describe circuits

that use only lines to define circuits. As shown by the data both macros had a high success rate with the exporting DXF macro having 100% success rate and the reference creating macro has a 100% success rate. This shows a good indication that both macros can handle a variety of shapes and dimensions and execute correctly.

Finishing and using the macros created sets up the inputs for the main process of creating multi-functional g-code. As seen in Figure 3.5 a reliable, accurate, and simple method of extracting information was developed for SolidWorks using its incorporated API capabilities. The macro code can be found in appendix C.

3.7 Slicer and Python

The next phase of the research focuses on creating a single g-code file that can be used in a multi-functional printer to print in a single process a multi-functional part. The first step after designing a part and exporting it from the CAD program is to import it into a slicer that will process the file and make a file that can be read by the printer. Although, there are several paid and open source slicers available they are not all suited for this projects intentions. The most popular slicers where compare at the time, as the more used and popular these software's are the better support and reliability they should have. The three main slicers we looked at were Slic3r, Simplify 3D, and CURA. Slic3r and CURA are both open source and Simplify 3D is paid for. All of the slicers support a vast majority of printers and if the printer you are using is not there you can easily add a profile for the printer and start using it. All of the slicers also support STL and AMF formats, and all have a large variety of settings to change. Much of the preference comes from the user to decide which one is better. For this project one of the test machines is based on the Tazbot from Lulzbot (Loveland, Coloroado) and they recommend CURA as their slicing software, furthermore CURA is written in python and allows for post process plugins to be developed in python that can run from inside CURA. Simplify 3D also allows some type of post processing in the form of scripting but that is very light and done in g-code.

One other development choice that has to be made before programing starts is what programming language will be used. There are two main requirements for the programming language, first it must be able to create a Graphical User Interface (GUI), and it must be able to handle math rather well. Some of the potential candidates where Java, C++, and Python, all of the languages support both requirements and some of the slicer programs share the same language. Slic3r is written in C++ and CURA in python, as for simplify 3D it is not open source therefore the source code is not available. As stated before CURA has the ability to run plugins and because it is programmed in python so do the plugins that run form it. CURA natively runs python. In this case if python is used it can seemingly be integrated into CURA for post-process development. For those reasons Python and CURA are chosen to start development of the multi-functional software.

3.8 Developing a Plugin for CURA

Developing software that can incorporate multi-functionality is the main goal yet it must be simple and straightforward for the user to use. This includes thinking about how the software will be installed and run, for this to be achieved understating how CURA works and implementing code that enhances the functionality without hindering usability is very important. Initially interfacing the plugin with CURA was easy, the user downloads and installs CURA then drops the python program inside the plugins folder and the plugin can be accessed through the CURA GUI. There is no need to compile just drag and drop the .py file into the plugins folder inside CURA. In an effort to separate the GUI from the slicer to make it future proof, in case of updates or other unforeseen circumstances and also the addition of libraries that are not supported by the python version in CURA. A program should not heavily rely on other software that it does not have control over. This is fixed by making a small simple plugin that runs the main GUI as an executable this allows CURA to change without affecting the main GUI, and if there are changes, the plugin is simple enough to quickly modify and update.

With more files needed to make the plugin work setup changes but not by much. A small plugin is places inside the plugin folder of CURA and with it a folder, inside this folder all the supplementary programs will be included allowing a straightforward and simple installation of the software into CURA. CURA interprets python programs a little different than normal, in general python executes the same, but a header composed of comments in the program is used to define variables and names for the plugin. Figure 3.7 shows the header code in the plugin and how it is shown in CURA.

```
1. #Name: America Makes II
2. #Info: Post process for multifunctionality
3. #Depend: GCode
```

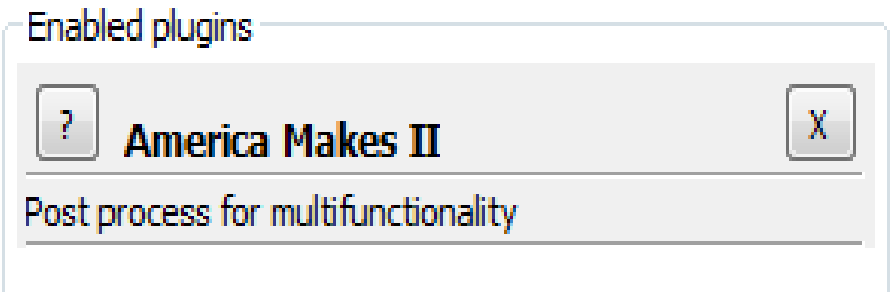


Figure 3.7 Python header file and how CURA process and displays it in the GUI

The plugin runs two executable files created from python. The first executable runs a small window that prompts the user for the location of the project. Once the location of the project is selected the program saves the generated g-code from CURA into the project folder with the name “temp.gcode”, this file only holds extrusion information for basic 3D Printing. Once the files have been saved it will execute the second program which is the main GUI that will create the multifunctional code. Figure 3.8 shows how the first pop up window looks that ask for the project location. As stated before this plugin is the interface between CURA and the multi-functional software. When the “Done” button is pressed the plugin grabs the input and passes it as an argument to the multi-functional program. The plugin code can be found below.

```
#Name: America Makes II
#Info: Post process for multifunctionality
#Depend: GCode
#Type: postprocess
'''''
__copyright__ = "Copyright (C) 2015 Efrain Aguilera - Released under my special terms"
import re
from Cura.util import profile
'''
from array import *
import os
import sys
import subprocess
import shlex
import ctypes

#Find directory (through GUI)
cmd = '''"''' + os.getcwd() + '''\\plugins\\AmSoftware\\dev\\dist\\dirMain.exe"'''

#runs DirMain.exe
#User browses to the project directory which becomes the output captured in "out"
out = subprocess.check_output(cmd, shell = True)

#save the gcode generated from CURA into the project folder as "temp.gcode"
with open(filename, "r") as f1:
    cLines=f1.readlines()
with open(out + "\\temp.gcode", "w") as f2:
    f2.write(''.join(cLines))

#run main GUI for multifunctionality passing the project path as an argument
cmd = '''"C:\\Users\\eaguilera5\\GoogleDrive\\AM_software\\test\\gui\\dist\\main.exe
''' + out + '''"'''
out = subprocess.check_output([cmd,out], shell = True)
```

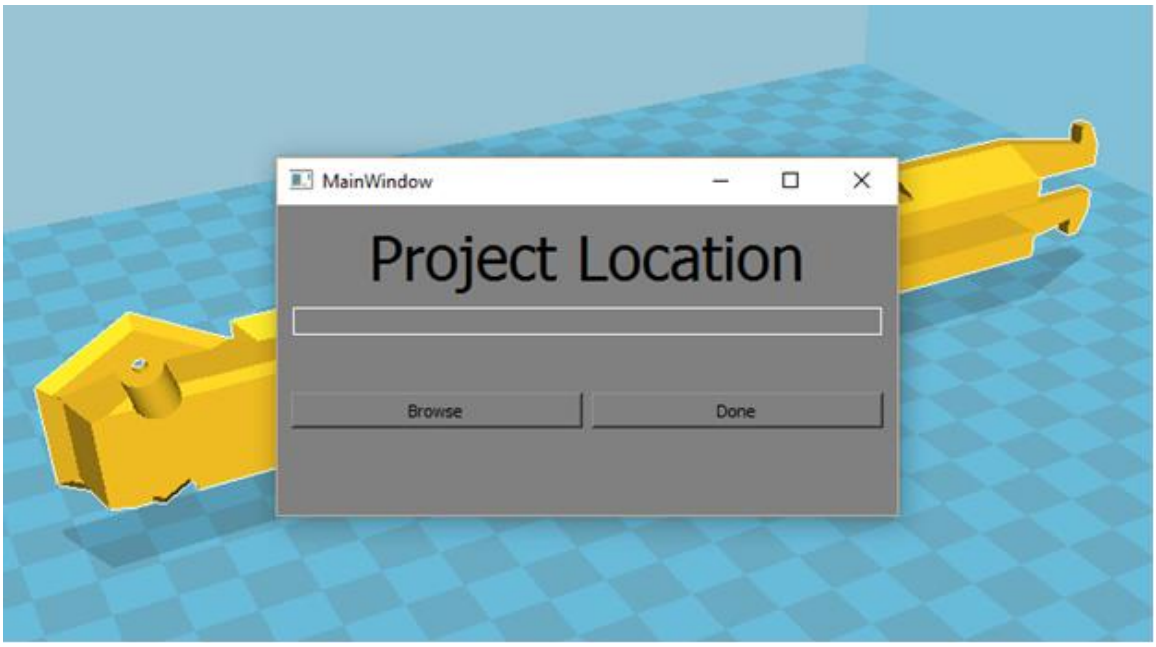


Figure 3.8 Screenshot of plugin initiating multi-functional software

3.9 Multi-Functional Software

The multi-functional GUI is static and allows the user to navigate several options and to verify the creation of the multi-functionality added. The multi-functional program has gone through several revisions the first of which is explained below. Subsequent version will also be shown an explained but the work has been done by others including Callum Bailey (Bailey, 2016).

The first version was the foundation and it includes only the basics for a functional circuit embedder. At the time of the first version of program the wire embedder which is the mechanical attachment that will allow the circuit to be physically be embedded into the 3D printed part was also being developed. Figure 3.9 illustrates how the first version of the program looks, the GUI is dived into two main parts. The left side of the GUI gives you control over the part, the top left lets you select the particular multi-functionality that needs to be added to the part, underneath there is a small box that lets the user select the project, if the program was executed through the plugin then the path will appear in the box and the user does not have to do anything. This allows

the user to change projects if the “temp.gcode” file has already been created and it exists in the project. Furthermore, the right side of the GUI holds verification information, this verification information is only visible after a multi-functionality is added to the part. As described before embedding circuits is the first functionality that is being added. The top right side of the GUI list the circuits that where embedded, clicking through the list will populate the bottom right corner with an image of the circuit for visual inspection.

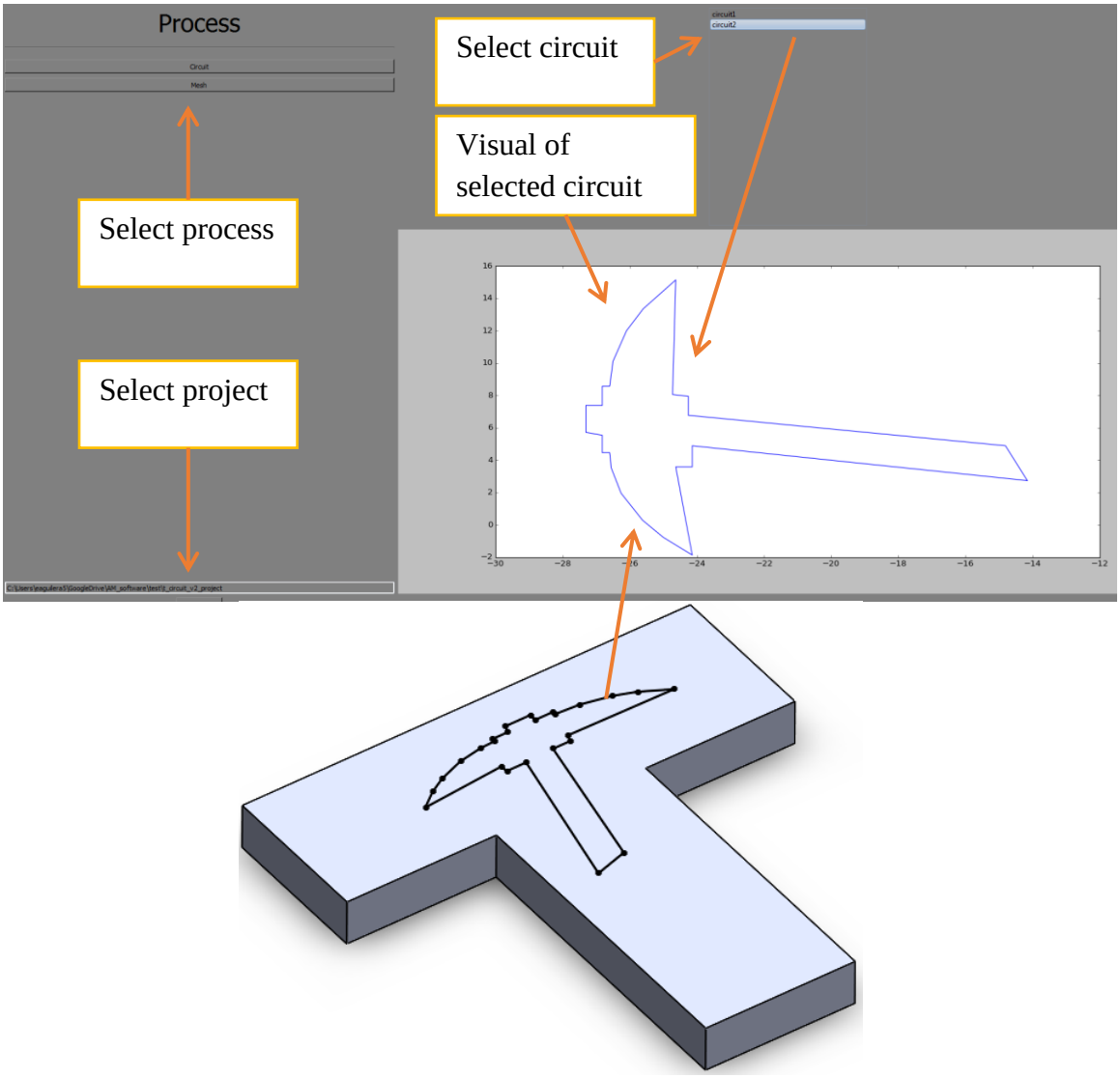
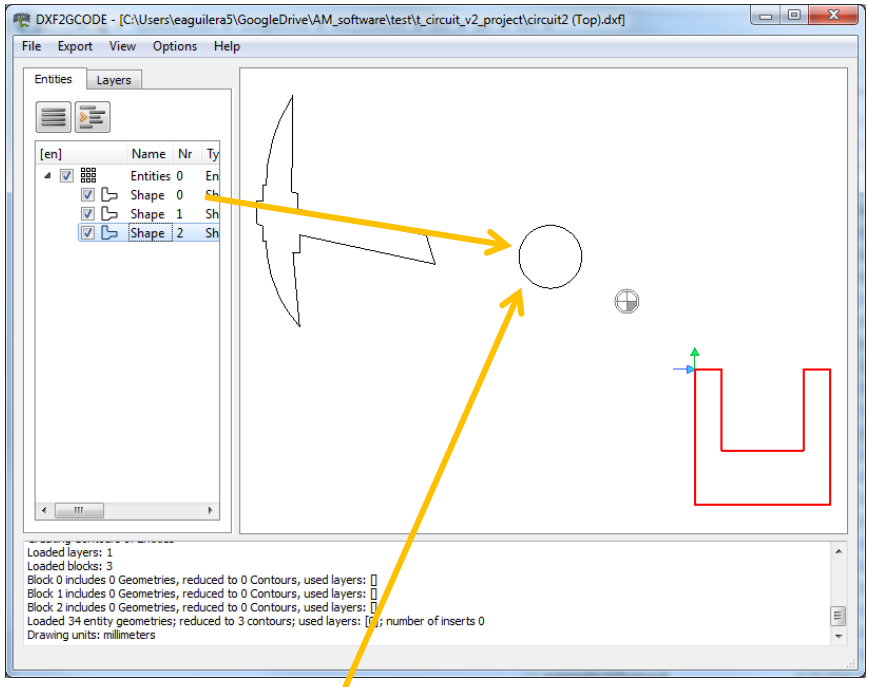


Figure 3.9 Multi-Functional GUI

The GUI is there for the user to add, verify, and navigate the multi-functionality capabilities. Under the “Process” column on the top left of the GUI a button labeled “circuit” is used to start the process of adding circuits to the part. Once the part has all the circuits embedded, the information is displayed in the right hand side and the user can traverse the list of circuits and inspect them visually. A final “final.gcode” file is created in the folder and that is what the multi-functional printer uses.

When the circuit button is clicked by the user, the function to add circuits is executed. The first task it performs is to read the “info.txt” and extract the number of circuits and their corresponding heights. The program then converts all of the DXF circuit files to g-code this is done by using an open source DXF to g-code converter called “dxf2gcode” although there are many programs that can do this, dxf2gcode was selected because it is written in python and can be easily modified and recompiled into an executable and it provides command prompt executions. This is important because the main program can use the command line to run the dxf2gcode converter in the background without any user intervention. A while loop goes through all the circuits DXF files and runs them through the dxf2gcode converter, once they are processed the new files with the extension “.ngc” are created in the project folder. The dxf2gcode converter arranges the circuit into sections described by the keyword “shape” as continuous segments. Figure 3.10 demonstrates what happens behind the scenes of dxf2gcode converter (a) shows the separate “shapes” and (b) shows how the g-code is exported. Once the while loop goes through all the circuits that need to be converter it exits and continues, leaving all of the g-code generated files inside the project folder.




```
(* SHAPE Nr: 0 *)
G0 X  -4.010 Y   4.917
M3 M8
G0 Z   3.000
F150
G1 Z  -1.500
F400
G2 X  -7.283 Y   1.644 I  -1.636 J  -1.636
G2 X  -4.010 Y   4.917 I   1.636 J   1.636
F150
G1 Z   3.000
G0 Z  15.000
M9 M5

(* SHAPE Nr: 1 *)
G0 X -14.153 Y   2.738
M3 M8
G0 Z   3.000
G1 Z  -1.500
F400
G1 X -24.139 Y   4.892
G1 X -24.139 Y   3.587
G1 X -24.630 Y   3.587
G1 X -24.139 Y  -1.859
G1 X -25.005 Y  -0.761
G1 X -25.621 Y   0.296
G1 X -26.260 Y   1.963
G1 X -26.553 Y   3.520
G1 X -26.600 Y   4.464
G1 X -26.821 Y   4.464
G1 X -26.821 Y   5.539
G1 X -27.309 Y   5.721
G1 X -27.309 Y   7.380
G1 X -26.821 Y   7.380
G1 X -26.821 Y   8.567
```

Figure 3.10 (a) Dxf2gcode converter separating continuous lines by shape (b) G-code generated by such file

Next, the “temp.gcode” file is open and loaded into an array this allows for quick and easy modification of the information stored in the file. The first thing done when “temp.gcode” is loaded is to find the reference shape in the first layer (layer 0). There is no way of knowing where the reference shape will be located both spatially and programmatically, yet CURA documents very well everything it creates. Using the key word “outer-wall” do differentiate the outer shell of each individual shape the program can go through the first layer and isolate every single wall-outer section. As it is going through the outer-wall blocks of code the reference shape is static so the number of coordinates is predictable at eight. Looking for the number of coordinates is a simple way of filtering through the different shapes and checking if it is the reference shape. Once the program finds a shape, with 8 sets of coordinates it extracts the x and y coordinates and compares them to a standard reference shape. The reference shape can be seen in Figure 3.11 this reference shape was picked for rotational accuracy as oppose to a square or circle that rotation would have no effect on. By having a reference shape that stays relative to the circuit and the part, then aligning the reference shapes also aligns the circuits to the part.

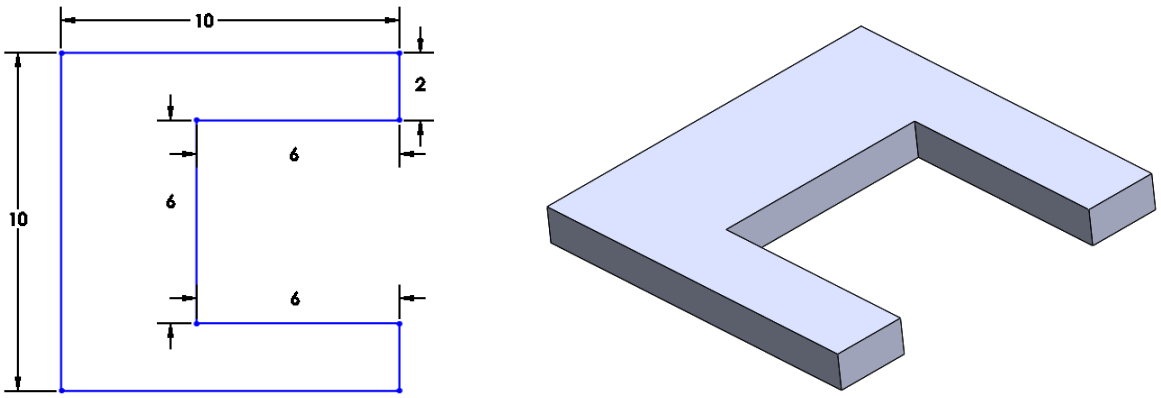


Figure 3.11 Reference shape used

The reference is compared using the Procrustes Analysis which will be explained below. One piece of information that is gathered from the analysis is the angle of rotation and the translation

in x and y which is saved in variables to be used later. After the reference shape is acquired and tested and a match is found it continues to the ".ngc" files created by the dxf2gcode which holds the circuit information but also holds a reference shape outline. This outline can be looked up in the same manner as before and once again using Procrustes Analysis the angle can be determined, for the circuits the translation does not need to be saved. Once the analysis is done the reference geometry on the circuit can be removed. At this point the difference in both angles, the one from the part and the one from the circuit can be calculated and that gives the total angle that the circuit must rotate to match into the part. The part is placed by the user therefore the circuits have to be the ones moved to match the location of the part. Once the rotation has been done it can be translated back into place. The translation coordinates used are generated from the parts translation and not the circuit, that is why the values of the translation of the circuit are not saved, they are not used. Once the rotation and translation are applied to the original coordinates of the circuit it is rewritten into the ".ngc" file with the correct g-code nomenclature as seen in figure 3.12.

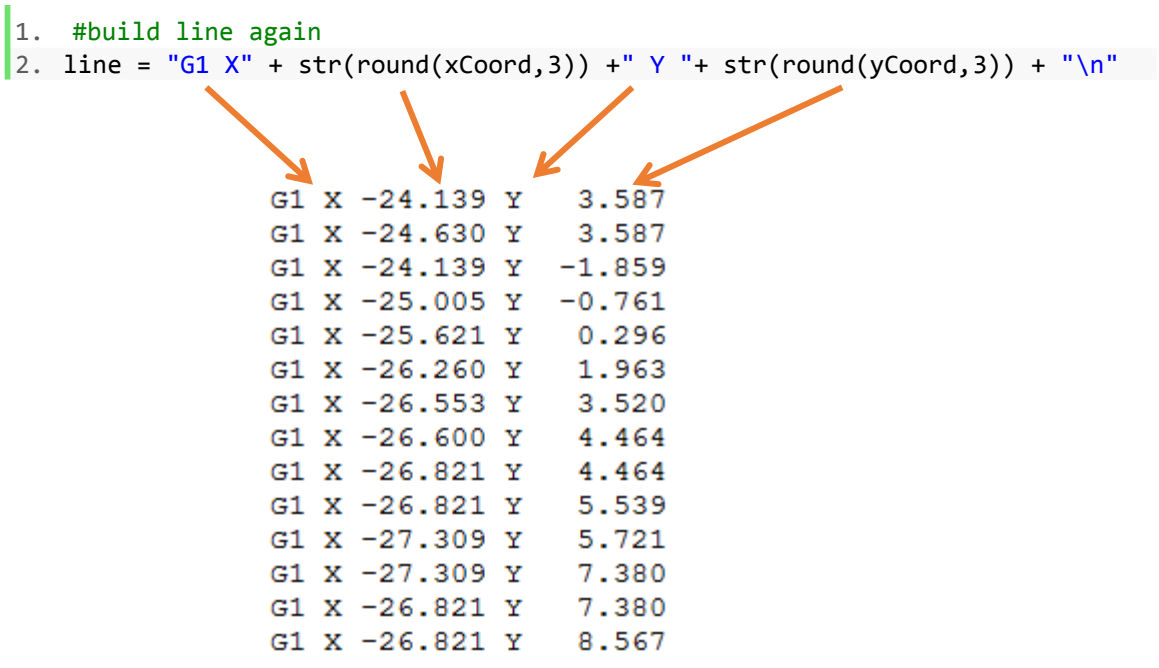


Figure 3.12 Code snippet to reproduce g-code for circuit

This is done to all of the circuit files, as they are corrected by their translation and rotation. Lastly, multi-functionality like wire embedding or machining is considered and add-on to the material deposition head. This leads to an extra offset to be placed because the project encompasses multiple tools to be used and hardcoding an offset is not a viable option. A small tool offset file is located in the directory this can at the moment be modified manually to update the tool offset as well as adding multiple tools all with different tools. The program and the file support multiple tool offsets, and example of what the file looks like is shown in Figure 3.13. Inside the code, the function getToolOffSet has an argument that specifies what tool, therefore inside the circuit embedding function the tool corresponds to 'tool1', for other processes the tool can be changed there without any hassle.

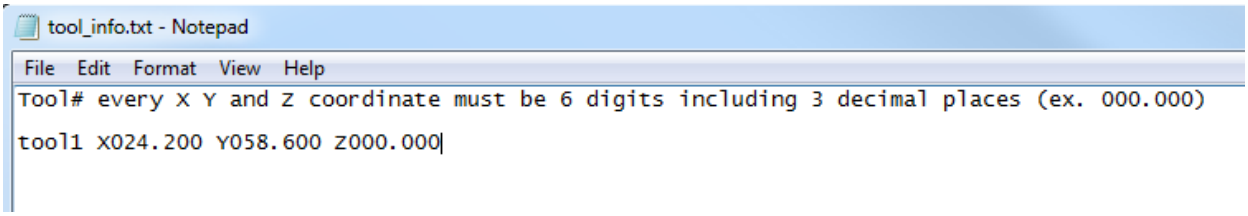


Figure3.13 Offset tool file to hold all of the tool offsets the multi machine could use

Once all of the circuits go through the process they are ready to be injected into the main g-code that holds the part information. The "temp.gcode" file is once again opened and this time it is parsed for layer height, as stated before CURA documents everything it does very well. Each layer is separated by a "LAYER" keyword and preceded by an x, y, and z command the z command is the height. By filtering through each layer and grabbing the z height it can be compared to the circuit z heights that were extracted at the beginning of the process. Once a match is found the circuit pertaining to the height can be inserted. With an effort to continue the naming convention that CURA offers a "Custom Circuit" block is created at the end of the layer

this is done at the end because the physical layer should be completed before it starts embedding the circuit. Figure 3.14 demonstrates the insertion of the custom circuit block at the end of the appropriate layer.

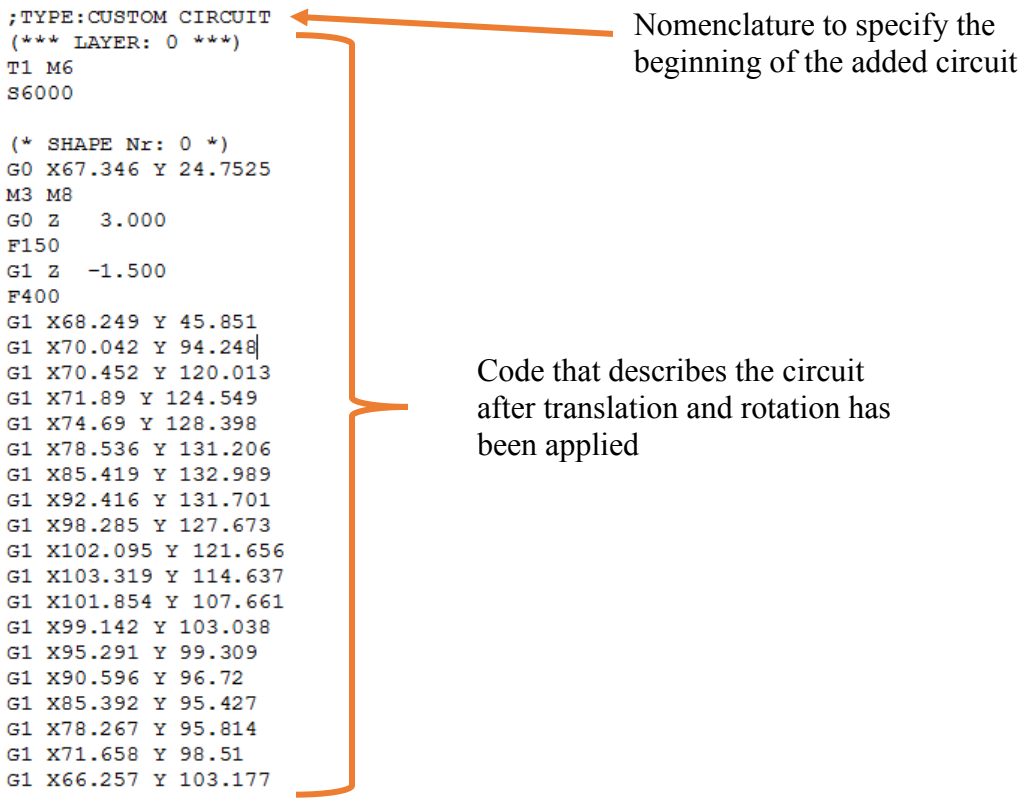


Figure 3.14 The DXF2gcode code with translation and rotation being inserted into the correct layer, note that the code is not condition to printing

Once the program has placed all the circuits in the correct height the program can finish the code and export a “final.gcode” file. It is important to note there is a key aspect missing in the code, the wire embedding code conditioning. Code conditioning for the wire embedder is where the code produced by the dxf2gcode converter is converted into usable g-code for the wire embedder. Due to the wire embedder being under development and the physical process not being tested, parameters for conditioning the g-code are unknown. Yet with the status of the program adding the conditioning can be easily included. As stated before multiple efforts are

being done and at the writing of this thesis the conditioning has been done by Callum Bailey (Bailey, 2016) and it will be described later in the thesis. As a proof of concept a simple UTEP pick was drawn into a cube, at this point no conditioning is done for wire embedding, yet the code was modified to work with a simple sharpie holder, this allows to test the tool offset, rotation and translation. Figure 3.15 shows the CAD drawing and then the printed part, the part was printed using a Lulzbot TAZ 5.

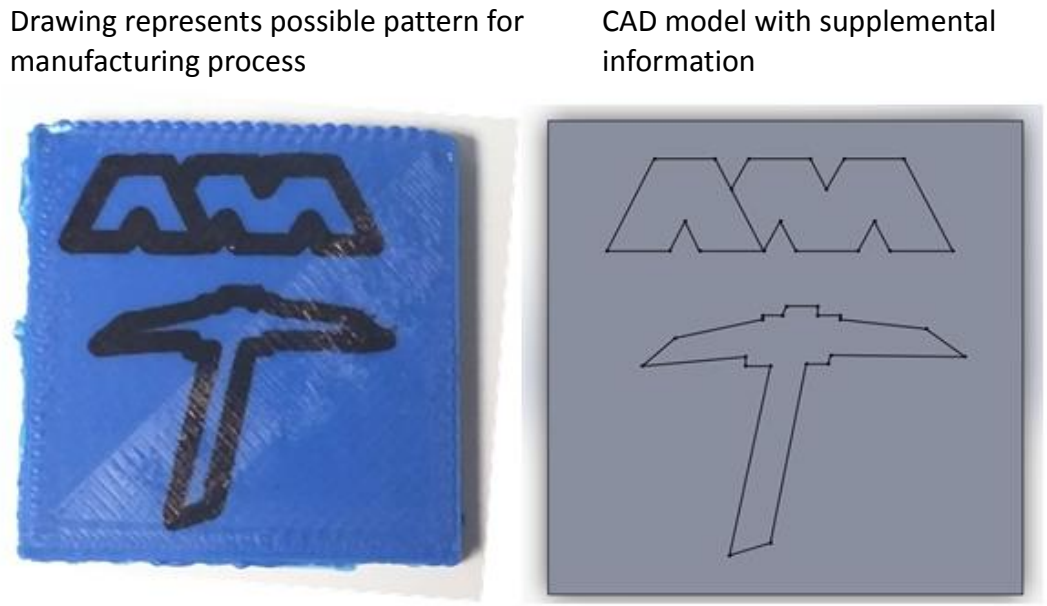


Figure 3.15 CAD design and printed part

3.10 Procrustes Analysis

When exporting the extra information from the SolidWorks model spatial data from the model to the circuit is lost. Using Procrustes analysis helps connect the lost information and relate the model to the circuit. The analysis can align models that are similar, different size, different position, and different rotation angle. The way this is performed is by starting with a reference position to be compared to. Since the model should always stay where the user places it is important for the model to be used as the reference. Rotation is done through the origin therefore the first step is to remove the translation. The translation equation is done first to place

the coordinates on the origin to allow for rotation and scaling to be performed. Equation 1 demonstrates the translation process for both the x coordinate and the y coordinate.

$$\overline{x} = \frac{x_1+x_2+\cdots x_n}{n} \qquad \overline{y} = \frac{y_1+y_2+\cdots y_n}{n}$$ Equation 1

When x-bar and y-bar are calculated the translation can be applied to each corresponding coordinate as seen in Equation 2.

$$x_i = x_i - \overline{x} \qquad y_i = y_i - \overline{x}$$ Equation 2

Next the rotation can be calculated using Equation 3, it is important to note that the rotation equation does depend on having one of the sets of coordinates be the reference. As one set of coordinates rotates to match the other, for this application our part coordinates will be static while the circuit coordinates will be rotated to match.

$$\theta = tan^{-1}\left(\frac{\sum_{i=1}^{l} w_i y_i - z_i x_i}{\sum_{i=1}^{l} w_i y_i + z_i x_i}\right)$$ Equation 3

When the angle is found the angle can be applied to each coordinate to get the rotated coordinate with Equation 4, assuming x_i and y_i are the reference coordinates.

$$(u_i, v_i) = (\cos(\theta)\, w_i - \sin(\theta)\, z_i\,,\ sin\,(\theta)\, w_i + \cos(\theta) z_i)$$ Equation 4

Scaling can also be achieved by using Equation 5 by finding a scale factor using the coordinates at the origin and then applying the scale factor to each as seen in Equation 6.

$$s = \sqrt{\frac{(x_i-\overline{x})^2+(y_i-\overline{y})^2+\cdots}{n}}$$ Equation 5

$$\left(\frac{x_i-\overline{x}}{s}, \frac{y_i-\overline{y}}{s}\right)$$ Equation 6

When all of the analysis are done a match test can be perform using Equation 7, the closer the result is to 1 the better match both shapes are. This is used to compare how accurate the analysis and it is used as a threshold to decide if the part has been correctly placed or not.

$$d = \sqrt{(u_i - x_i)^2 + (v_i - y_i)^2 + \cdots} \quad \text{Equation 7}$$

Using Procrustes Analysis the original translation and the angle can be computed using the reference shapes on the model and circuit, then used on the actual circuit to embed.

3.11 Component Placement

Once basic functionality of the multi-functional software was shown it was passed onto Callum (Bailey, 2016) for additional enhancements. As a complete package, the user should be able to create a multi-functional part in a CAD program and easily export and print it. Although basic support for exporting the needed information is already supported improvements are being worked on, this section goes in detail on what is being done to help the user embed circuits into a part inside SolidWorks.

A small macro is being developed that guides the user into creating 3D printed electronics. The macro pops up a GUI that allows the user to continue modifying the model while interfacing with the GUI. Figure 3.16 shows the GUI that pops up to help the user which can be invoked by a button on the toolbar for easy access. This GUI is all-inclusive with the reference shape generator and the export functionality and other additional services. A list of the buttons and text boxes are listed below with their role.

- Create Reference Button – This creates the reference shape inside your model, a popup comes up letting the user know to move the reference shape outside the model if it is inside.
- Export Button – Runs the export algorithm described before going through each layer and exporting the circuit information and height
- Browse Button – Allows the user to browse for a schematic file from EAGLE

- Schematic File Box – The box shows the location of the schematic filed selected through the browse button
- Load Components Button – Goes in and parses through the schematic file to grab all of the information that solid work needs for each component.
- Component Box – An updated list of the components that have not been placed on the part
- Route Button- This button is not functional at the moment more details in the future work section
- Exit Button –Exits the macro/GUI
- Place Button – When a plane is selected on the model and a component is selected on the component list, it will place the component on the plane.

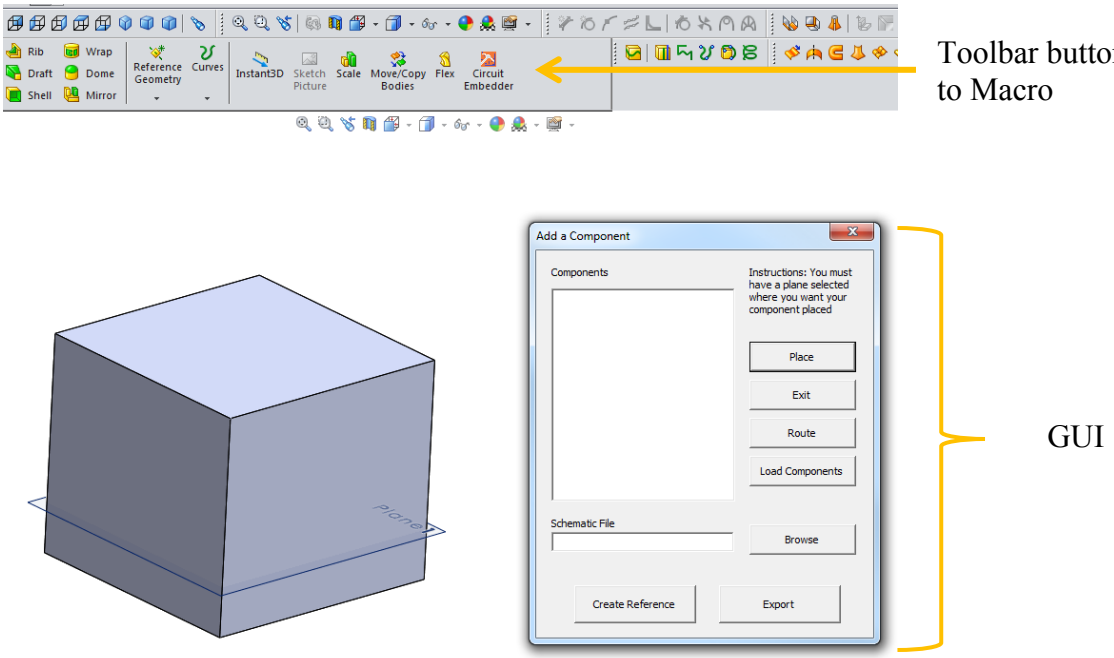


Figure 3.16 SolidWorks GUI for adding components

When the macro is launched, a GUI appears, the user has the option to browse to an EAGLE schematic and load components based on the schematic file. Due to the nature of the process, an

average schematic file created through existing EAGLE libraries lack information for AM parts including an outside perimeter and a height. Therefore a modified EAGLE library for AM is being populated the main changes are the inclusion of a perimeter layer and an attribute of height. A comparison of a classic component in eagle versus one in the modified library is shown in Figure 3.17.

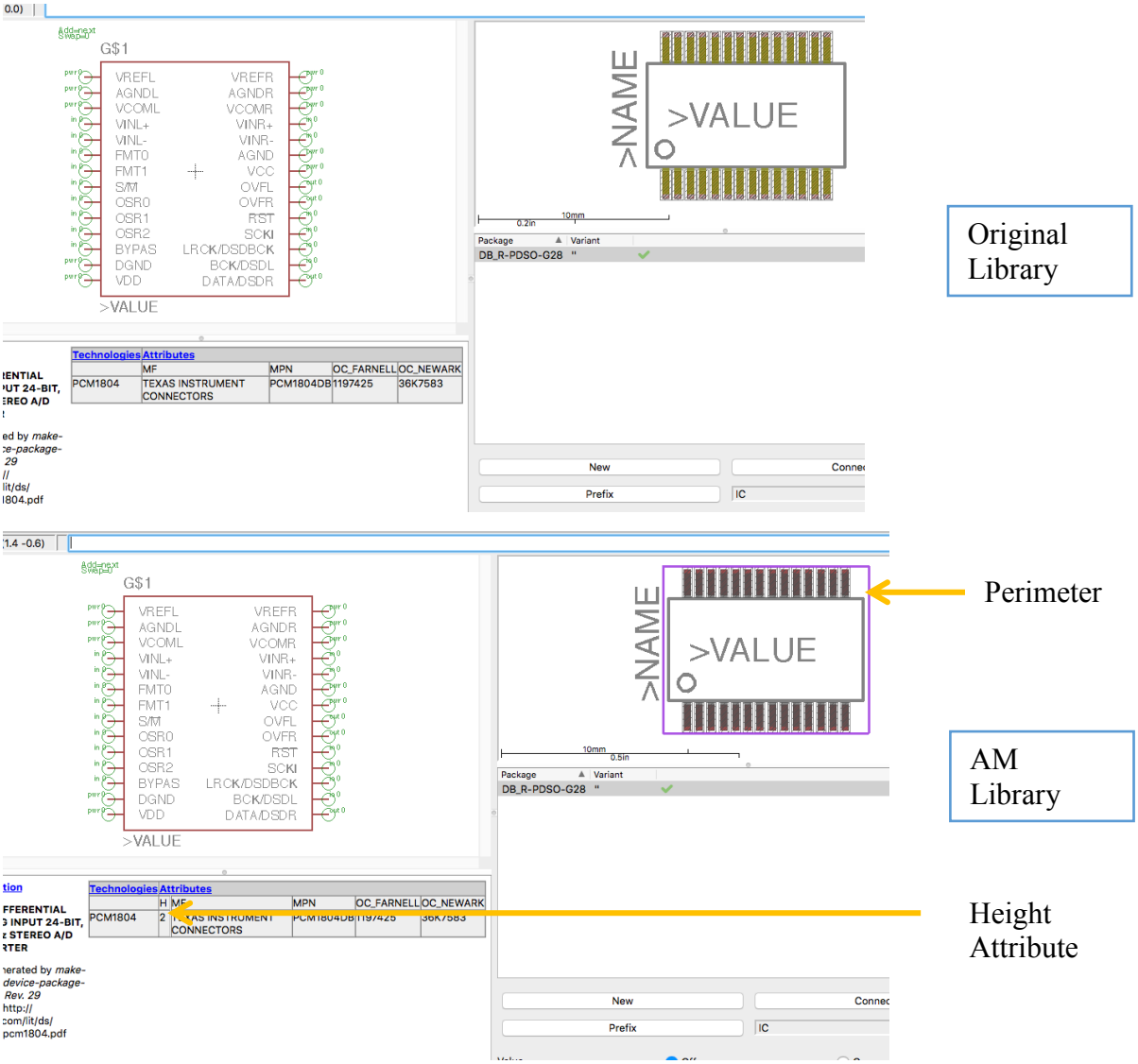


Figure 3.17 EAGLE library comparison to that of the modified library for AM

The lack of VBA functionality to parse XML made reading EAGLE schematic troublesome and time consuming therefore a simple parsing script was developed in python that is executed through VBA. This parsing script goes through the schematic and extracts all the necessary information like component names, perimeter, and height. Then it is placed in a simple text file that VBA can go through and extract information from. The text file is once again placed in the project folder of the part. At this moment, the parser only supports perimeters defined with lines. Once the parser exports all of the information needed to a text file VBA can go through it, extract the names, and compare them to the sketches inside the model part. By comparing the sketches to the name of the components, the macro can identifies which components have already been placed and omit those from the list. This also allows the user to pause or resume the creation of the part without having to completely add all of the components and if the user decides to delete a component it will show up again when the list is reloaded. Figure 3.8 shows a part with embedded components from the GUI. The naming convention of the added parts do not include “circuit#” and are not made in 3D sketches because the component cavities are part of the model therefore it is exported as a normal STL or AMF file. When the component is placed on a plane the user can move the sketch around and position it where it needs to be.

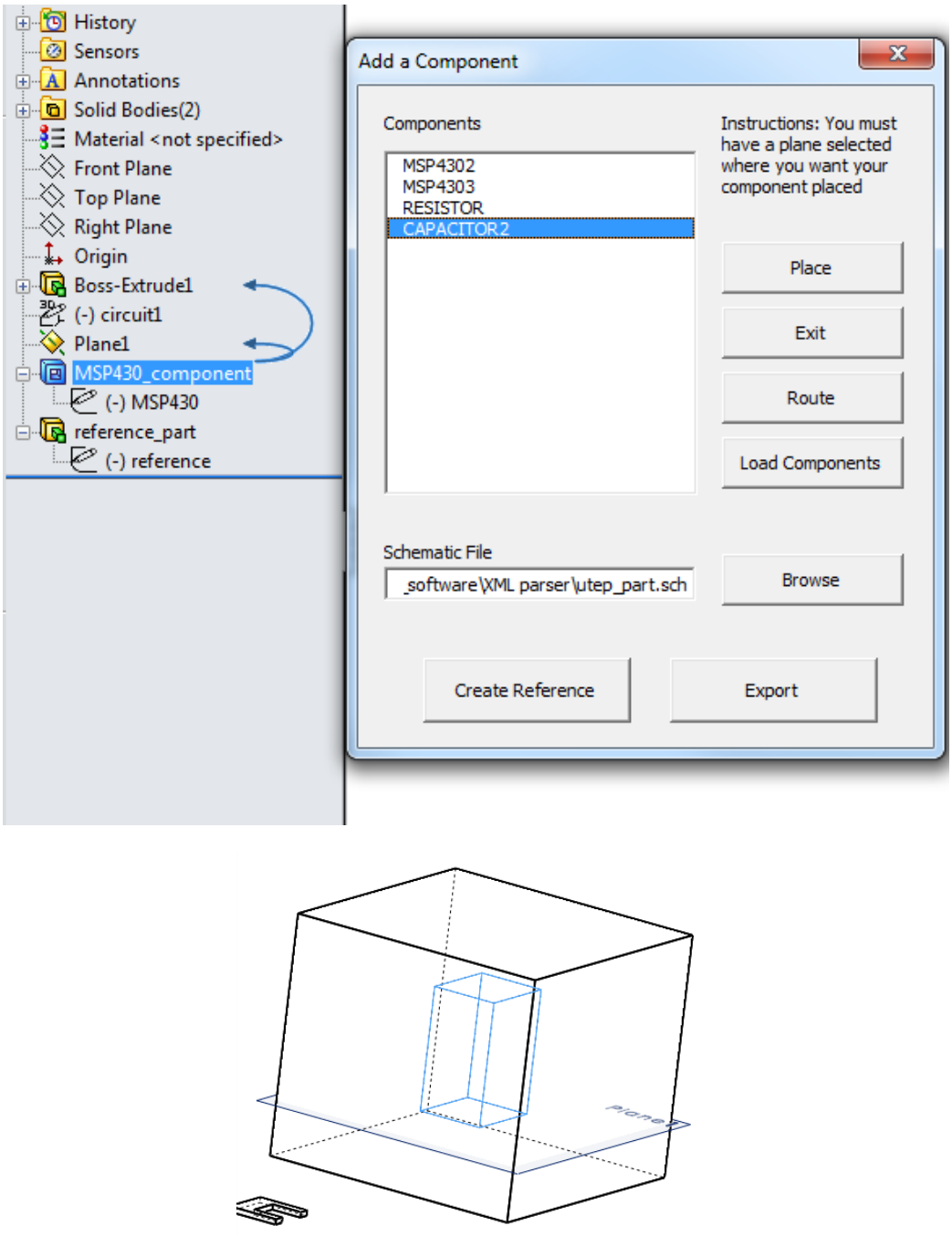


Figure 3.18 Component placement example using the developed macro

Once all of the components have been placed the user can manually go in and route the wires using the previous methods described before, either by hand or importing a DXF file from an EAGLE PCB.

3.12 Continuing Work and State of the Software

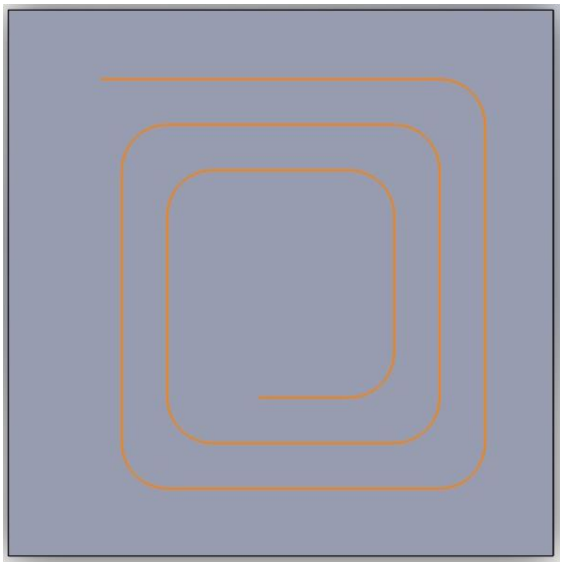
A team of students have been working on various parts of the project including continuations of what has been presented in this thesis, below is a summary of were the software package stands.

Callum (Bailey, 2016) continued with enhancements to the main program for integration of circuits into 3D printed parts. As the hardware became more mature and a process was establish a set of parameters was developed. Some of these parameters include tool initialization to select the proper tool to use the wire embedder or the extruding head, mechanical toolpaths and control requirements, and conditioning of the g-code to match the process parameter. For code conditioning some of the included process conditioning is listed below:

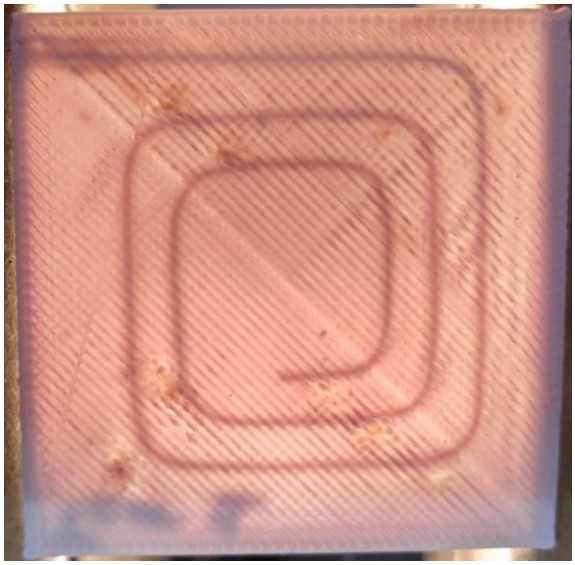
- Tool Rotation
- Staking (of the wire)
- Tapering
- Cooling
- Cutting (wire)

G-code supports G2 and G3 commands for arcs but for better control on wire embedding an algorithm to convert G2 and G3 commands to G1 and G0 commands was developed. Other work done by Bailey includes a foundation layer generation for parts that do not have a 100% fill and at the moment a dense 100% fill layer is needed to embed wire. This algorithm detects where to change the fill rate on a layer and creates a dense layer below a circuit. Further work was done in error catching and making the software more user friendly and robust. Work by Callum (Bailey, 2016) includes adding a comprehensive list of parameters that the user has access to modify things like offsets and tools, maximum angle to turn and the number of strokes to cut the wire,

the changes are included in the final g-code. Figure 3.19 shows what a completed piece using the developed software looks like, the part has a square coil embedded in the center of the piece, It was designed in SW and exported using the created macro, then sliced through CURA and processed through the multi-functional program which outputted a single g-code file that was used with the multi-functional machine.



CAD Drawing of Test Piece



Actual Test Piece

Figure 3 3.19 A CAD drawing and a finished part designed and processed through the developed code

4.0 Conclusion and Future Work

4.1 Conclusion

A software package for the creation of multi-functional g-code was demonstrated. A macro that interfaces with SolidWorks allows the user to easily implement components into the part that is being design with the help of a professional schematic tool (EAGLE). The macro also allows the extraction of information that the software package needs to create multi-functional parts. With the automation of reference shapes and output of circuit layers.

Furthermore, a process to import and process the information exported from SolidWorks was establish. The process directly interfaces with CURA a slicing software that is being used to process the part for user friendliness. Launching the main GUI from within the CURA slicing software makes a smooth transition from the slicing software to the multi-functional software. Using Procrustes Analysis the spatial information that is lost through export of SolidWorks can be recreated. Allowing a spatially accurate circuit to be embedded within the model.

Additionally, concurrent work done by other members of the group helped shape and clean up the process. With the addition of g-code conditioning to correctly move and embed wire, the ability to modify a range of parameters, capabilities to process G2 and G3 commands, create 100% fill layers to embed in, and error handling for a better user experience.

Lastly an effort to make a simple software was deployed, to install the software the user only needs to have the normal software installed, SolidWorks, CURA, and EAGLE. A single folder and a plugin file have to be moved in to the plugin folder of CURA. Inside SolidWorks a single macro button can be added to the tool bar that points to macro located in the original folder inside the CURA plugins folder.

4.2 Future Work

All though a large amount of work has been done in the project there is still details missing. Inside the GUI a better parameter handling algorithm is being developed and as the embedding process is perfected different parameters are being changed or added. Inside the GUI

there is a section of real state reserved for the visualization of the processed files. An effort to make a 3D and 2D comprehensive view that includes specific wire embedding commands for the user to see and as a way of verifying the design. Furthermore, an increase of error handling to increase the robustness and ease of use of the program for easier debug and better error messages for the user.

Within the Component Placement macro inside SolidWorks the routing option is being investigated. With the capability of reading the netlist from a schematic the user could be able auto route inside SolidWorks or have an easier time routing connections on each layer. Like with the main program error handling and robustness of the macro is also being improved to create a more pleasant user experience.

In the hardware there is development on other multi-functionality like foil embedding an routing that can ultimately be added to the software package. The software at this point has been design as a backbone that can handle multiple processes, wire embedding has been at the core of it and will help develop the other process, as much of the process is similar apart from the specific code conditioning. There is also a custom machine being developed that is not based on the Lulzbot models that will include multi-functionality. This software should be able to interface several machines as long as the machines are g-code base.

References

Aguilera,E., Ramos, J., Espalin, D., Maestas E., Cedillos, F., Muse, D., MacDonald, E., and Wicker, R.B., ”3D Printing of Electro Mechanical Systems”, 24th International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference, Austin, TX, (2013).

Ahlers, D., “Development of a Software for the Design of Electronic Circuits in 3D-Printable Objects” Bachelor Thesis, at the Research Group Technical Aspects of Multimodal Systems, TAMS, (2015).

ASTM Standard F2792-12a. 2012. “Standard Terminology for Additive Manufacturing Technologies”. ASTM International, West Conshohocken, PA.

C. Bailey, ”G-Code Generation for Multi-Process 3D Printing”, Master’s Thesis, University Texas at El Paso, 2016.

Cham J., Pruitt B., Cutkosky M., Binnard M., Weiss L., Neplotnik G. “Layered manufacturing with embedded components: process planning considerations”. Proceedings of DETC99: 1999 ASME Design Engineering Technical Conference, (1999).

DeNava, E., Navarrete, M., Lopes, A., Alawneh, M., Contreras, M., Muse, D., Castillo, S., MacDonald, E., Ryan Wicker. “Three-Dimensional Off-Axis Component Placement and Routing for Electronics Integration using Solid Freeform Fabrication”. In Proceedings of Solid Freeform Fabrication Symposium, the University of Texas at Austin, Austin TX, (2008).

Espalin, D., Herrera, L., Hossain, M. S., & Wicker, R. “Wire reinforced 3D printed parts made by using material extrusion additive manufacturing process”. *In* CAMX 2014 - Composites and

Advanced Materials Expo: Combined Strength. Unsurpassed Innovation, The Composites and Advanced Materials Expo (CAMX), (2014).

Espalin, D., Muse, D.W., MacDonald, Wicker, R.B. "3D Printing multifunctionality: structures with electronics", The International Journal of Advanced Manufacturing Technology. Springer London, (2014).

Gibson, I., Rosen, D.W., Stucker, B. Additive manufacturing technologies: Rapid prototyping to direct digital manufacturing. New York: Springer. (2010)

Hiller, J.,Lipson, H., "STL 2.0: A PROPOSAL FOR A UNIVERSAL MULTI-MATERIAL ADDITIVE MANUFACTURING FILE FORMAT ", In Proceedings of Solid Freeform Fabrication Symposium, the University of Texas at Austin, Austin TX, (2009).

ISO/ASTM 52915:2013

KUMAR, V. & DUTTA, D. "An assessment of data formats for layered manufacturing". Advances in Engineering Software, (1997).

Kruth J-P, Leu MC, Nakagawa T . "Progress in additive manufacturing and rapid prototyping". CIRP Ann Manuf Technol, (1998).

Kataria A, Rosen DW." Building around inserts: methods for fabricating complex devices in stereolithography". 2000 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Baltimore, MD, (2001).

Lopes, A.J., Navarrete, M., Medina, F., Palmer, J., MacDonald, E. and Wicker, R.B. “Expanding rapid prototyping for electronic systems integration of arbitrary form”. 17th Annual International Solid Freeform Fabrication Symposium, Austin, TX (2006).

Medina, F., Lopes, A.J., Inamdar, A.V., Hennessey, R., Palmer, J.A., Chavez, B.D. and Wicker, R.B.” Integrating multiple rapid manufacturing technologies for developing advanced customized functional devices”. Rapid Prototyping & Manufacturing 2005 Conference Proceedings, Rapid Prototyping Association of the Society of Manufacturing Engineers, Dearborn, Michigan., (2005).

Medina, F., Lopes, A.J., Inamdar, A.V., Hennessey, R., Palmer, J.A., Chavez, B.D., Yang P., Gallegos O.L., and Wicker, R.B., “Hybrid Manufacturing: Integrating Direct Write and Stereolithography,” 16th Annual Solid Freeform Fabrication Symposium, University of Texas at Austin, (2005).

National Aeronautics and Space Administration (NASA) (2016) Outgassing data for selecting spacecraft materials. Retrieved from http://outgassing.nasa.gov/

Navarrete, M., Lopes, A.J., Acuna, J., Estrada, R., MacDonald, E., Palmer, J., Wicker, R.B.” Integrated Layered Manufacturing of a Novel Wireless Motion Sensor System with GPS” 18th International Solid Freeform Fabrication Symposium - An Additive Manufacturing Conference, Austin, TX (2007).

Richard I Olivas, "Conformal electronics packaging through additive manufacturing and micro-dispensing" (January 1, 2011). *ETD Collection for University of Texas, El Paso.* Paper AAI1494367. http://digitalcommons.utep.edu/dissertations/AAI1494367

Shemelya, C.M., Zemba M., Liang, M., Espalin, D., Kief, C., Xin, H., Wicker, R.B., MacDonald, E. 3D printing multi-functionality: Embedded RF antennas and components," 2015, 9th European Conference on Antennas and Propagation (EuCAP), Lisbon.

Tröger, C., Bens, A.T., Bermes, G. "Ageing of acrylate-based resins for stereolithography: thermal and humidity ageing behavior studies". Rapid Prototype, (2008).

"Home." *Voxel8*. N.p., n.d. Web. 02 Dec. 2016.

Wang, T., Xi, J., Jin, Y. "A model research for prototype warp deformation in the FDM process". International Journal of Advanced Manufacturing Technology 33, (2007).

Wasserfall, F., "Embedding of SMD populated circuits into FDM printed object" In Proceedings of Solid Freeform Fabrication Symposium, the University of Texas at Austin, Austin TX, (2015).

Weiss L, Prinz F,.Novel applications and implementations of shape deposition manufacturing. Naval Research Reviews, Office of Naval Research, (1998).

Wohlers, T.T.,Wohlers report 2016, rapid prototyping, tooling and manufacturing: state of the industry. Annual Worldwide Progress Report, Wohlers Associates Inc., Fort Collins, CO., (2016).

Appendix-A

PLUGIN CODE

```
#Name: America Makes II
#Info: Post process for multifunctionality
#Depend: GCode
#Type: postprocess
'''
__copyright__ = "Copyright (C) 2015 Efrain Aguilera -
 Released under my special terms"
import re
from Cura.util import profile
'''
from array import *
import os
import sys
import subprocess
import shlex
import ctypes

#Find directory (through GUI)
cmd = '''"''' + os.getcwd() + '''\\plugins\\AmSoftware\\dev\\dist\\dirMain.exe"'''
#runs DirMain.exe
#User browses to the project directory which becomes the output captured in "out"
out = subprocess.check_output(cmd, shell = True)

#save the gcode generated from CURA into the project folder as "temp.gcode"
with open(filename, "r") as f1:
    cLines=f1.readlines()
with open(out + "\\temp.gcode", "w") as f2:
    f2.write(''.join(cLines))

#run main GUI for multifunctionality passing the project path as an argument
cmd = '''"C:\\Users\\eaguilera5\\GoogleDrive\\AM_software\\test\\gui\\dist\\main.exe ''
' + out + '''"'''
out = subprocess.check_output([cmd,out], shell = True)
```

DIRECTORY MAIN

```
#Name: GUI
#Info: Post process for multifuntionality
#Depend: GCode
#Type: postprocess
__copyright__ = "Copyright (C) 2015 Efrain Aguilera - Released under terms of me"
import re
import sys
from PyQt4 import QtGui, QtCore
from Cura.util import profile
import dirGui
import os
import subprocess
import ctypes

#import plugDev.py

#GUI controls
class window(QtGui.QMainWindow, dirGui.Ui_MainWindow):
```

```
    def __init__(self):
        super(self.__class__,self).__init__()
        self.setupUi(self)
        self.btnBrowse.clicked.connect(self.browse_folder)
        self.btnSave.clicked.connect(self.saveExit)

    def browse_folder(self):
        self.lineEdit.setText(QtGui.QFileDialog.getExistingDirectory(self, "Select Project"))

    def saveExit(self):
        createfile(str(self.lineEdit.text()))
        sys.exit()

def main():
    app = QtGui.QApplication(sys.argv)
    form = window()
    form.show()
    app.exec_()
    #create file for
def createfile(dirFile):
    line = dirFile

    print os.getcwd()
    with open(os.getcwd() + "\\tempinfo.txt","w") as f:
        f.write(line)

main()
```

XML PARSER

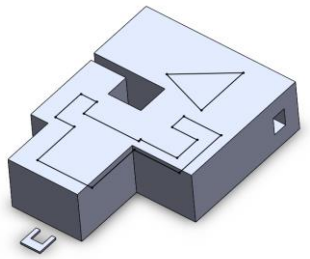
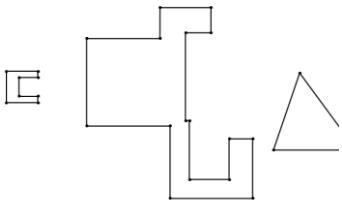
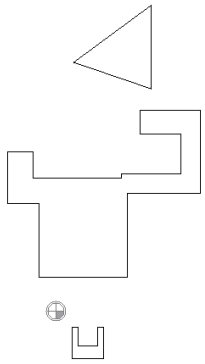
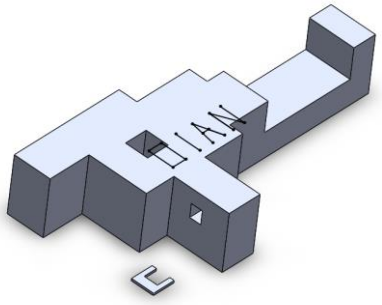
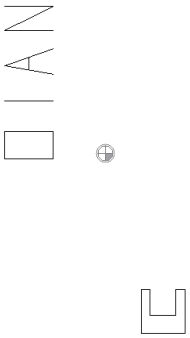
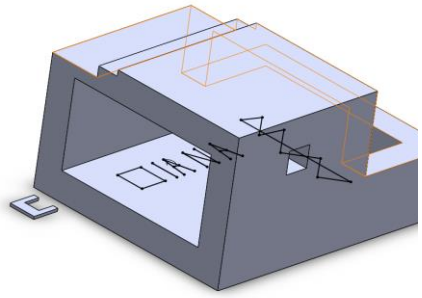
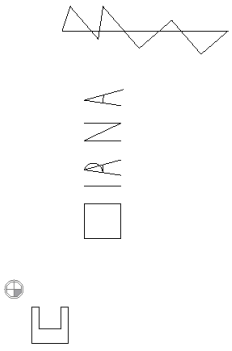
```
#Author: Efrain Aguilera
#UTEP
#Version 1 10/21/2016
    #reads a schematic file and outputs a text file
    #with the list of components and its perimeters
#Version 2 10/26/2016
    #made procrustes analysis its own function
    #get coordinates for pads or smd connections
    #
    #Supports
    #Perimeter - Wire
    #connections- Pads and SMD



import sys
import os
import xml.etree.ElementTree
import ctypes

global partList
global packageList
global partInfo
global deviceList
global partPerimeter
partList = []
deviceList = []
partPerimeter = []
global e
```

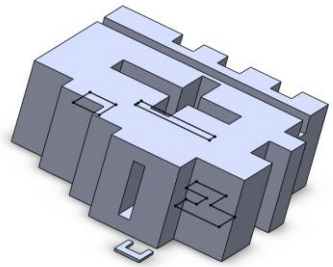
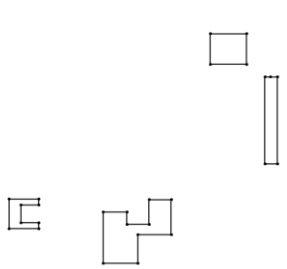
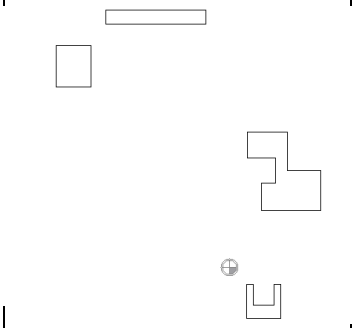
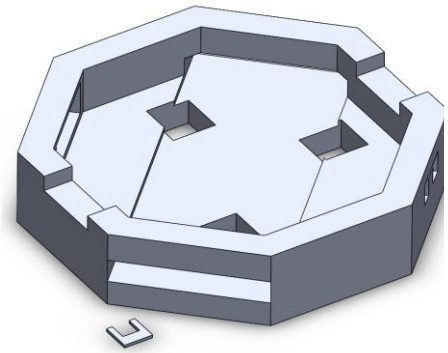
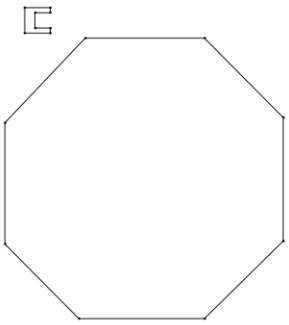
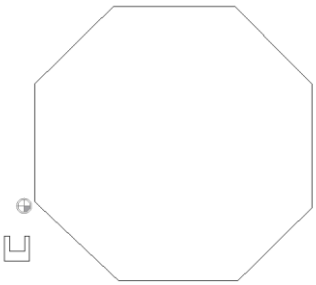
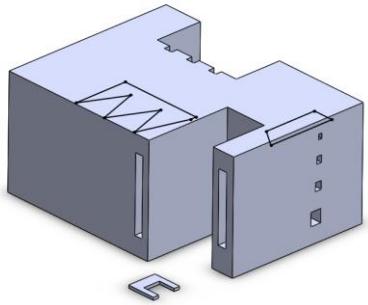
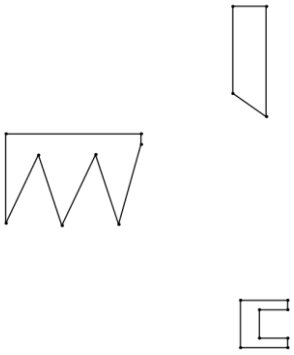
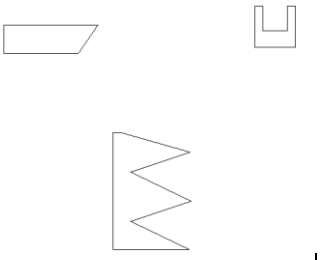
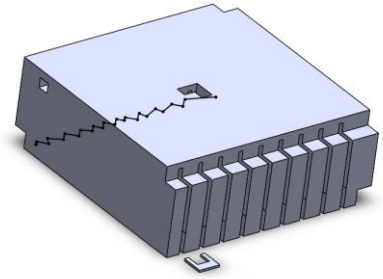
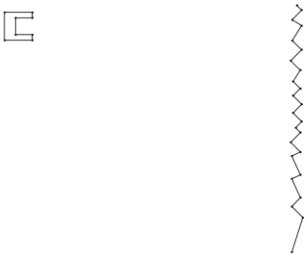
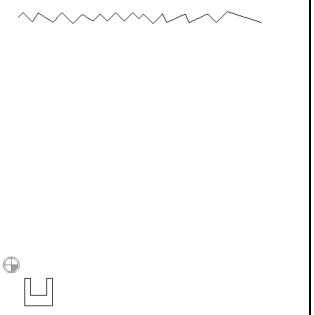
```
fPath = sys.argv[1]
dirName = sys.argv[2]
e = xml.etree.ElementTree.parse(fPath).getroot()
#initialize file
infoFile = dirName + "/sch_info.txt"
with open(infoFile,"w") as f:
    f.close
def getParts():
    global partList
    for part in e.iter('part'):
        partList.append(part.get('deviceset'))

def getDevice():
    global deviceList
    for device in e.iter('part'):
        deviceList.append(device.get('device'))
def createTextFile():
    #prepare arrays
    for i in range(0,len(partList)):
        partList[i] = partList[i]+"\n"
    for i in range(0,len(deviceList)):
        deviceList[i] = deviceList[i]+"\n"
    #write text file
    with open("sch_info.txt","a") as f:
        f.write(''.join(partList))
        f.close()
    with open("sch_info.txt","a") as f:
        f.write(''.join(deviceList))
        f.close()
def getPerimeterOrigin(x1,x2,y1,y2):
    x = []
    y = []
    for i in range(0, len(x1)):
            x.append(float(x1[i]))
            x.append(float(x2[i]))
    for i in range(0, len(y1)):
        y.append(float(y1[i]))
        y.append(float(y2[i]))
    if len(x) != 0:
        nx = []
        ny = []
        xbar = sum(x)/len(x)
        ybar = sum(y)/len(y)
        for i in x:
            nx.append(i - xbar)
        for i in y:
            ny.append(i - ybar)
    else:
        nx = []
        ny = []
        xbar = 0
        ybar = 0
    return nx,ny,xbar,ybar

def getPerimeterandPads(layer):
    name = []
    for part in e.iter('part'):
        name.append(part.get('device'))
    count = 0
    for part in e.iter('package'):
        if part.get('name') == name[count]:
```

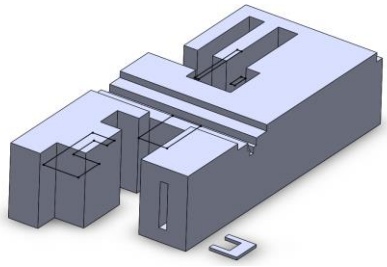
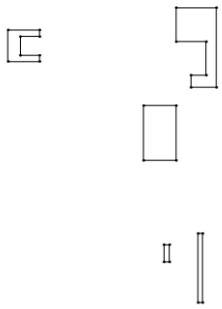
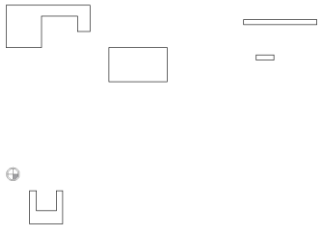
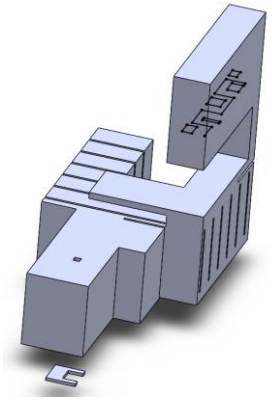
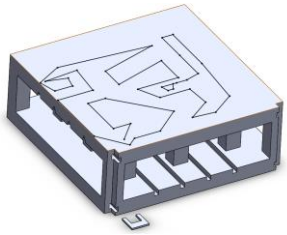
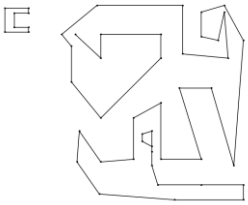
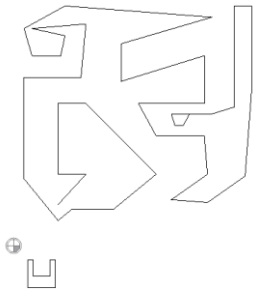
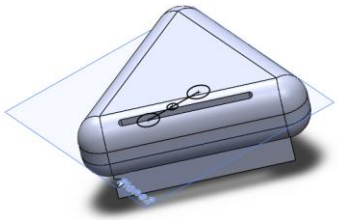
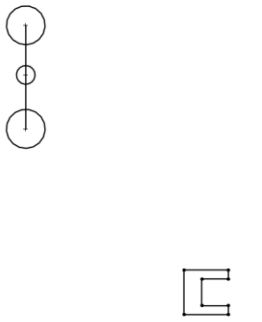
```
            x1 = []
            x2 = []
            y1 = []
            y2 = []
            x = []
            y = []
        #look for 'wire' with a layer of Perimeter
            for crd in part.iter("wire"):
                if int(crd.get('layer')) == int(layer):
                            x1.append(crd.get('x1'))
                            x2.append(crd.get('x2'))
                            y1.append(crd.get('y1'))
                            y2.append(crd.get('y2'))
                    x,y,xbar,ybar= getPerimeterOrigin(x1,x2,y1,y2)
                    #prepare  coordinate arrays for file
                    coordinates = []
                    for i in range(0,len(x)):
                        coordinates.append(str(x[i]) + ',' + str(y[i]) + '\n')
                    #clear arrays to reuse and create the rest
                    x = []
                    y = []
                    dx = []
                    dy = []
                    padX = []
                    padY = []
                    #get coordinates for smd
                    for crd in part.iter("smd"):
                        x.append(crd.get('x'))
                        y.append(crd.get('y'))
                        dx.append(crd.get('dx'))
                        dy.append(crd.get('dy'))
                    # find center of smd
                    for i in range(0,len(x)):
                        deltaX = float(dx[i])/2
                        deltaY = float(dy[i])/2
                        padX.append(float(x[i])+deltaX)
                        padY.append(float(y[i])+deltaY)
                        #translate to origin
                        padX[i] = padX[i] - xbar
                        padY[i] = padY[i] - ybar

                    #prepare   smd coordinate arrays for file
                    smdCoordinates = []
                    for i in range(0,len(x)):
                        smdCoordinates.append(str(padX[i]) + ',' + str(padY[i]) + '\n')

                    x = []
                    y = []
                    #get coordinates for pads
                    for crd in part.iter("pad"):
                        x.append(float(crd.get('x')))
                        y.append(float(crd.get('y')))
                    #translate to origin
                    for i in  range(0,len(x)):
                        x[i] = x[i] - xbar
                        y[i] = y[i] - ybar
                    #prepare   smd coordinate arrays for file
                    padCoordinates = []
                    for i in range(0,len(x)):
                        padCoordinates.append(str(x[i]) + ',' + str(y[i]) + '\n')
                    #grab height
```

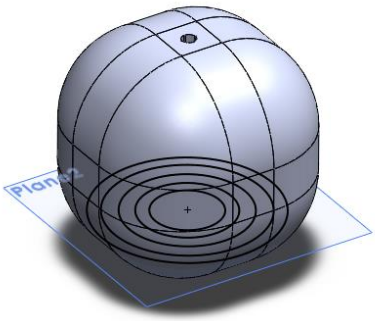
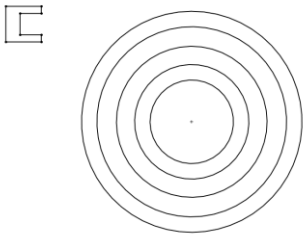
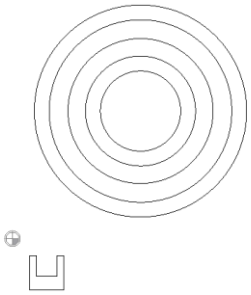
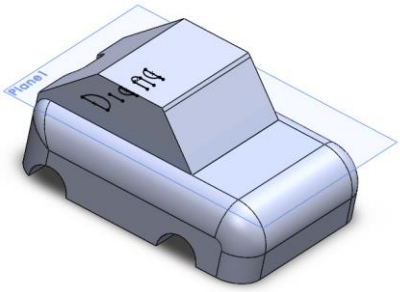
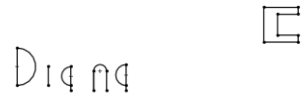
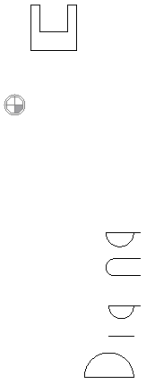
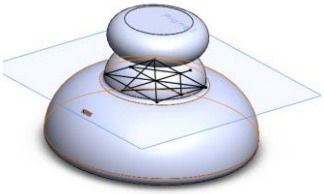
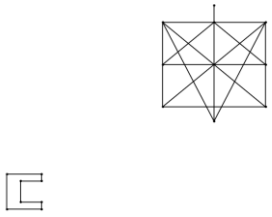
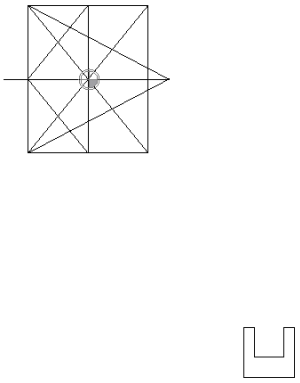
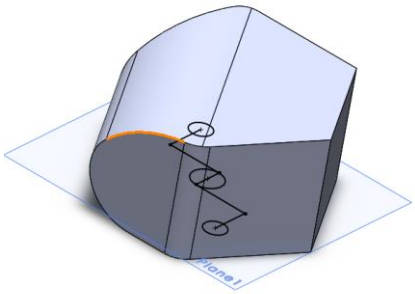
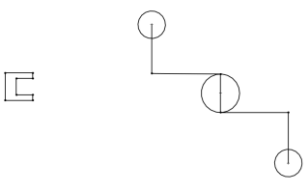
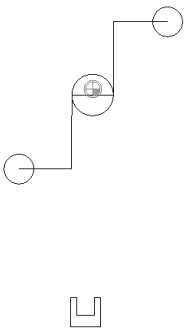
```
                    h= grabHeight(name[count])
                    #open file to add information
                    with open(infoFile,"a") as f:
                        f.write('part\n')
                        f.write(name[count] + '\n')
                        f.write('h='+ h + '\n')
                        f.write('PStart\n')
                        f.write(''.join(coordinates))
                        f.write('PEnd\n')
                        f.write('smdStart\n')
                        f.write(''.join(smdCoordinates))
                        f.write('smdEnd\n')
                        f.write('padStart\n')
                        f.write(''.join(padCoordinates))
                        f.write('padEnd\n')
                        count +=1
                        if count == len(name):
                            break
        def getPLayer():
            # find the layer number that is designated to the 'Perimeter'
            position = 0
            layerName = []
            layerNumber = []
            for layer in e.iter('layer'):

                layerName.append(layer.get('name'))
                layerNumber.append(layer.get('number'))
            for i in range(0,len(layerName)):
                if layerName[i] == 'Perimeter':
                    position = i
                    break
            return layerNumber[position]
        def grabHeight(partName):
            h = 'NA'
            for part in e.iter('device'):
                if part.get('name') == partName:
                    for att in part.iter("attribute"):
                        if att.get('name') == "H":
                            h = att.get("value")
            return h

        pLayer = getPLayer()
        getPerimeterandPads(pLayer)
```

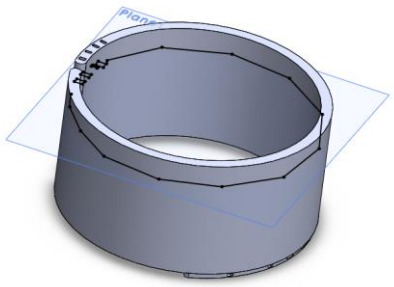
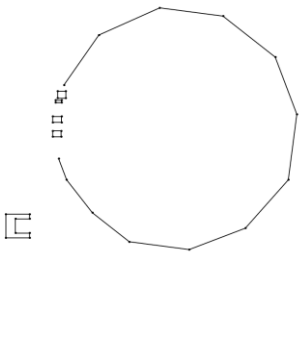
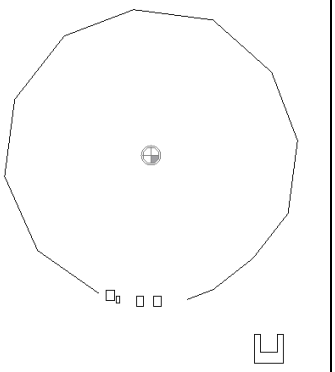
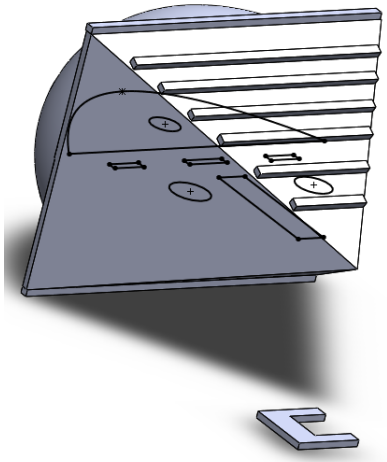
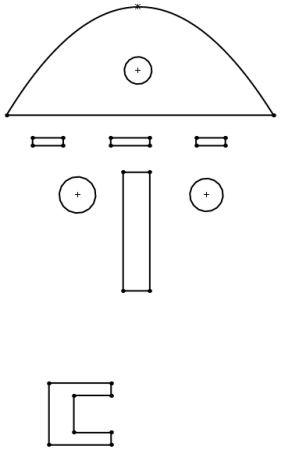
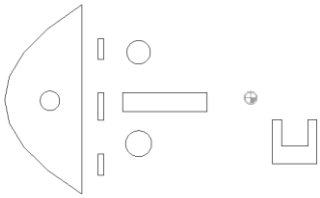
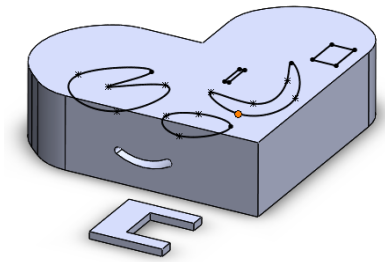
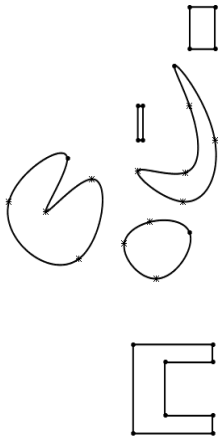
Appendix-B

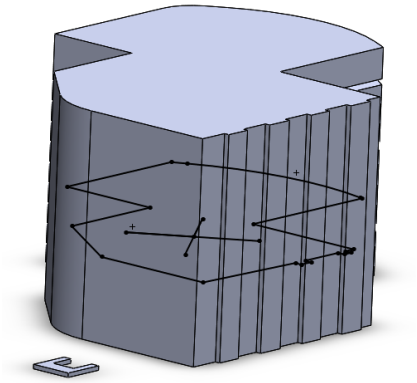
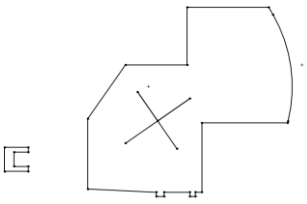
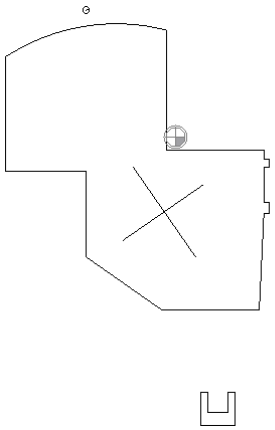
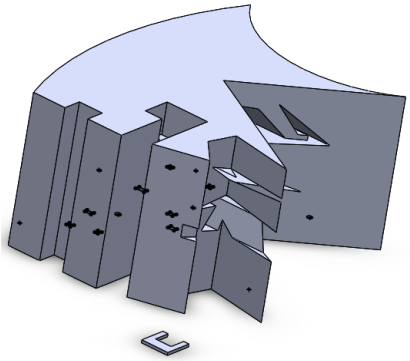
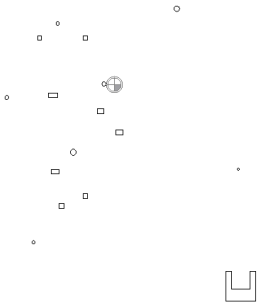
Type	SolidWorks	Export	DXF
Line 1			
Line 2			
Line 3			

Line 4			
Line 5			
Line 6			
Line 7			

Line 8			
Line 9			
Line 10			
Curve 1			

Curve 2			
Curve 3			
Curve 4			
Curve 5			

Curve 6			
Curve 7			
Curve 8			

Curve 9			
Curve 10			

Appendix-C

MACRO GUI CODE

```
Dim swApp As Object

Dim Part As Object
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long
Dim swSketchMgr As SldWorks.SketchManager
Dim vSkLines As Variant
Dim swSketch As SldWorks.Sketch
Dim swFeat As SldWorks.Feature

Private Sub exportDXF_Click()
    'Variable Declarations
    'Option Explicit
    Dim swApp As Object
    Dim fs As Object
    Dim f As Object
    Dim infoName As String
    Dim swPart As Object
    Dim sModelName As String
    Dim sPathName As String
    Dim varAlignment As Variant
    Dim dataAlignment(11) As Double
    Dim varViews As Variant
    Dim dataViews(1) As String
    Dim options As Long
    Dim boolstatus As Boolean
    Dim sketchName As String
    Dim i As Integer
    Dim dxfPathName As String
    Dim dirName As String
    Dim dxfName As String
    Dim zValue As Double
    Dim sketchPointArray As Variant
    Dim sManager As SldWorks.SelectionMgr
    Dim ft As SldWorks.Feature
    Dim theSketch As SldWorks.Sketch
    Dim pointCount As Long
    Dim sViewData As Object
    Dim myFeatMr As SldWorks.FeatureManager
    Dim deleteOption As Long
    Dim longstatus As Long

    'Definitions
    Set swApp = Application.SldWorks
    Set swPart = swApp.ActiveDoc
    Set sManager = swApp.ActiveDoc.SelectionManager
    'Setting paths, grabs the location of the part file
    sModelName = swPart.GetPathName
    sPathName = swPart.GetPathName
    sPathName = Left(sPathName, Len(sPathName) - 7)
    dirName = sPathName + "_project"
    dxfPathName = sPathName
    infoName = sPathName
    sPathName = sPathName + ".dxf"
    'Creates array for aligment to be used in export of DXF
```

```
    dataAlignment(0) = 0#
    dataAlignment(1) = 0#
    dataAlignment(2) = 0#
    dataAlignment(3) = 1#
    dataAlignment(4) = 0#
    dataAlignment(5) = 0#
    dataAlignment(6) = 0#
    dataAlignment(7) = 1#
    dataAlignment(8) = 0#
    dataAlignment(9) = 0#
    dataAlignment(10) = 0#
    dataAlignment(11) = 1#

    varAlignment = dataAlignment
    'Select top view for export
    dataViews(0) = "*Top"
    varViews = dataViews
    'Hide all bodies
    boolstatus = swPart.Extension.SelectByID2("Solid Bodies", "BDYFOLDER", 0, 0, 0, Fal
se, 0, Nothing, 0)
    swPart.FeatureManager.HideBodies
    'Show reference
    boolstatus = swPart.Extension.SelectByID2("reference", "SKETCH", 0, 0, 0, False, 0,
 Nothing, 0)
    swPart.UnblankSketch
    ' create folder
    Set fs = CreateObject("Scripting.FileSystemObject")
    If Dir(dirName, vbDirectory) = "" Then
        fs.CreateFolder (dirName)
    End If
    ' create file
    Set f = fs.CreateTextFile(dirName + "\info.txt")
    i = 1
    sketchName = "circuit" + CStr(i)
    boolstatus = swPart.Extension.SelectByID2(sketchName, "SKETCH", 0, 0, 0, False, 0,
Nothing, 0)
    Set fs = CreateObject("Scripting.FileSystemObject")
    ' Make all sketches invisible
    Do While boolstatus
        swPart.BlankSketch
        i = i + 1
        sketchName = "circuit" + CStr(i)
        boolstatus = swPart.Extension.SelectByID2(sketchName, "SKETCH", 0, 0, 0, False,
 0, Nothing, 0)
    Loop
    'Make each sketch visible and export them seperately
    i = 1
    sketchName = "circuit" + CStr(i)
            boolstatus = swPart.Extension.SelectByID2(sketchName, "SKETCH", 0, 0, 0, Fal
se, 0, Nothing, 0)
            Do While boolstatus
                swPart.UnblankSketch
                swPart.Extension.SelectByID2 sketchName, "SKETCH", 0, 0, 0, False, 0, No
thing, 0
                Set ft = sManager.GetSelectedObject4(1)
                Set theSketch = ft.GetSpecificFeature
                sketchPointArray = theSketch.GetSketchPoints
                pointCount = UBound(sketchPointArray) + 1
                'Getting z height
                zValue = Round(sketchPointArray(1).Y, 4)
                f.WriteLine ("<" + sketchName + "_" + CStr(zValue) + ">")
```

```
111.                dxfName = dirName + "\" + sketchName + ".dxf"
112.                'Export each annotation view to a separate drawing file
113.                swPart.ExportToDWG2 dxfName, sModelName, 3, False, varAlignment, False,
    False, 0, varViews
114.                swPart.Extension.SelectByID2 sketchName, "SKETCH", 0, 0, 0, False, 0, No
    thing, 0
115.                swPart.BlankSketch
116.                i = i + 1
117.                sketchName = "circuit" + CStr(i)
118.                boolstatus = swPart.Extension.SelectByID2(sketchName, "SKETCH", 0, 0, 0,
     False, 0, Nothing, 0)
119.
120.            Loop
121.            f.Close
122.            'Show all bodies
123.            boolstatus = swPart.Extension.SelectByID2("Solid Bodies", "BDYFOLDER", 0, 0,
     0, False, 0, Nothing, 0)
124.            swPart.FeatureManager.ShowBodies
125.            'Hide reference
126.            boolstatus = swPart.Extension.SelectByID2("reference", "SKETCH", 0, 0, 0, Fa
    lse, 0, Nothing, 0)
127.            swPart.BlankSketch
128.            'Debug.Print "Inspect DXF files in " + Left(sPathName, Len(sPathName) -
     16)
129.        End Sub
130.        Private Sub createReference_Click()
131.            Dim swApp As Object
132.
133.            Dim Part As Object
134.            Dim boolstatus As Boolean
135.            Dim longstatus As Long, longwarnings As Long
136.            Dim swSketchMgr As SldWorks.SketchManager
137.            Dim vSkLines As Variant
138.            Dim swSketch As SldWorks.Sketch
139.            Dim swFeat As SldWorks.Feature
140.
141.            Set swApp = _
142.            Application.SldWorks
143.
144.            Set Part = swApp.ActiveDoc
145.            boolstatus = Part.Extension.SelectByID2("Top Plane", "PLANE", 0, 0, 0, False
    , 0, Nothing, 0)
146.
147.            Set swSketchMgr = Part.SketchManager
148.
149.            swSketchMgr.InsertSketch True
150.            Set swSketch = Part.GetActiveSketch2
151.            Set swFeat = swSketch
152.            swFeat.Name = "reference"
153.            Part.ClearSelection2 True
154.            'Creates a 5mm reference
155.            'swSketchMgr.CreateLine -0.01, -0.01, 0, -0.01, -0.005, 0
156.            'swSketchMgr.CreateLine -0.01, -0.005, 0, -0.005, -0.005, 0
157.            'swSketchMgr.CreateLine -0.005, -0.005, 0, -0.005, -0.006, 0
158.            'swSketchMgr.CreateLine -0.005, -0.006, 0, -0.007, -0.006, 0
159.            'swSketchMgr.CreateLine -0.007, -0.006, 0, -0.007, -0.009, 0
160.            'swSketchMgr.CreateLine -0.007, -0.009, 0, -0.005, -0.009, 0
161.            'swSketchMgr.CreateLine -0.005, -0.009, 0, -0.005, -0.01, 0
162.            'swSketchMgr.CreateLine -0.005, -0.01, 0, -0.01, -0.01, 0
163.
164.            'Creates a 10mm reference
```

```
            swSketchMgr.CreateLine -0.015, -0.015, 0, -0.015, -0.005, 0
            swSketchMgr.CreateLine -0.015, -0.005, 0, -0.005, -0.005, 0
            swSketchMgr.CreateLine -0.005, -0.005, 0, -0.005, -0.007, 0
            swSketchMgr.CreateLine -0.005, -0.007, 0, -0.011, -0.007, 0
            swSketchMgr.CreateLine -0.011, -0.007, 0, -0.011, -0.013, 0
            swSketchMgr.CreateLine -0.011, -0.013, 0, -0.005, -0.013, 0
            swSketchMgr.CreateLine -0.005, -0.013, 0, -0.005, -0.015, 0
            swSketchMgr.CreateLine -0.005, -0.015, 0, -0.015, -0.015, 0

            Part.InsertSketch2 True
            Set swFeat = Part.FeatureManager.FeatureExtrusion3(True, False, False, 0, 0,
     0.001, 0.001, False, False, False, False, False, False, False, False, False, False, Fa
    lse, False, True, 0, 0, False)
            boolstatus = Part.SelectedFeatureProperties(0, 0, 0, 0, 0, 0, 0, 1, 0, "refe
    rence_part")

            MsgBox "Please move reference sketch/feature outside of your part"
            boolstatus = Part.Extension.SelectByID2("reference", "PLANE", 0, 0, 0, False
    , 0, Nothing, 0)

        End Sub

        Private Sub BrowseButton_Click()

            Dim Filter                      As String

            Dim fileName                    As String

            Dim fileConfig                  As String

            Dim fileDispName                As String

            Dim fileOptions                 As Long
            Set swApp = _
                Application.SldWorks
            strFileToOpen = swApp.GetOpenFileName("Please choose a file to open", "", Fi
    lter, fileOptions, fileConfig, fileDispName)
            AddComponent.XMLFile = strFileToOpen

        End Sub

        Private Sub ExitButton_Click()
            Unload AddComponent
        End Sub


        Private Sub PlaceButton_Click()
        Dim component As String
        Dim x1, x2, y1, y2, h As Double
        Dim swFeature As Object
        Dim swFeatureManager As FeatureManager


        Set swApp = _
            Application.SldWorks
            Set Part = swApp.ActiveDoc
            'boolstatus = Part.Extension.SelectByID2("Top Plane", "PLANE", 0, 0, 0, Fals
    e, 0, Nothing, 0)
            Dim i As Integer
            'Go through the list and find the component to selected to place
            For i = 0 To ComponentsList.ListCount - 1
```

```
                If ComponentsList.Selected(i) Then
                    component = ComponentsList.list(i)
                End If
            Next i
            'get selected plane
            Dim swSketchMgr As SldWorks.SketchManager
            Dim vSkLines As Variant
            Dim swSketch As SldWorks.Sketch
            'search for perimeter
            Set swSketchMgr = Part.SketchManager
            swSketchMgr.InsertSketch True
            Set swSketch = Part.GetActiveSketch2
            Set swFeat = swSketch
            swFeat.Name = component
            Part.ClearSelection2 True
            i = 0
            Do While i < arraySize
                If data(i) = component Then
                    'grab height and skip to perimeter
                    h = getHeight(i + 1)
                    i = i + 2
                    Do While data(i + 1) <> "PEnd"
                        x1 = getX(i + 1)
                        y1 = getY(i + 1)
                        x2 = getX(i + 2)
                        y2 = getY(i + 2)
                        swSketchMgr.CreateLine x1, y1, 0, x2, y2, 0
                        i = i + 2
                    Loop
                    'extrude sketch
                    Set swFeature = Part.FeatureManager.FeatureCut3(True, False, False,
    0, 0, h, h, False, False, False, False, 0, 0, False, False, False, False, False, False,
     True, True, True, False, 0, 0, False)

                    'Dim FeatureName As System.String
                    boolstatus = Part.SelectedFeatureProperties(0, 0, 0, 0, 0, 0, 0, Tru
    e, False, component + "_component")
                End If
                i = i + 1
            Loop
            LoadComponents_Click
        End Sub

        Private Sub LoadComponents_Click()
            '-----------------------------------
            'get a list of components
            '-----------------------------------

            Dim fileName As String
            Dim dataLine As String
            Dim pos As Integer
            Dim fileLength As Integer
            Dim oXMLFile As Object
            Dim XMLatt As String
            Dim i As Long
            Dim lPosition As Long
            Dim line As String
            Dim bool As Boolean

            Set swApp = Application.SldWorks
            Set swPart = swApp.ActiveDoc
```

```
278.            Set sManager = swApp.ActiveDoc.SelectionManager
279.
280.            fileName = strFileToOpen
281.            'runs script to create sch_info.txt file
282.            Dim fs As Object
283.            Dim sPathName As String
284.            Dim macroPath As String
285.            Dim dirName As String
286.            Dim wsh As Object
287.
288.            Set wsh = VBA.CreateObject("WScript.Shell")
289.            Dim waitOnReturn As Boolean: waitOnReturn = waitOnReturn
290.            Dim windowStyle As Integer: windowStyle = 0
291.
292.            sPathName = swPart.GetPathName
293.            sPathName = Left(sPathName, Len(sPathName) - 7)
294.            dirName = sPathName + "_project"
295.
296.            ' create folder
297.            Set fs = CreateObject("Scripting.FileSystemObject")
298.            If Dir(dirName, vbDirectory) = "" Then
299.                fs.CreateFolder (dirName)
300.            End If
301.            macroPath = swApp.GetCurrentMacroPathName
302.            macroPath = Left(macroPath, Len(macroPath) - 13)
303.            wsh.Run macroPath & "\xmlParser\xml_parser_v2.exe " & fileName & " " & dirNa
    me, 0, True
304.            fileName = dirName + "\sch_info.txt"
305.
306.            'open file
307.            Open fileName For Input As #1
308.            'find length of file in lines
309.            i = 0
310.            Do Until EOF(1)
311.                Line Input #1, dataLine
312.                i = i + 1
313.            Loop
314.            Close #1
315.            arraySize = i
316.            ReDim data(arraySize)
317.            'clear list box
318.            ComponentsList.Clear
319.            'fill array with information
320.            Open fileName For Input As #1
321.            i = 0
322.            Do Until EOF(1)
323.                Line Input #1, dataLine
324.                data(i) = dataLine
325.                i = i + 1
326.            Loop
327.            Close #1
328.            'get parts name
329.            For i = 0 To arraySize
330.                If data(i) = "part" Then
331.                    boolstatus = swPart.Extension.SelectByID2(data(i + 1), "SKETCH", 0,
    0, 0, False, 0, Nothing, 0)
332.                    'loop through sketches to make sure it has not been placed before
333.                    If boolstatus = False Then
334.                        AddComponent.ComponentsList.AddItem data(i + 1)
335.                    End If
336.                End If
```

```
            Next i

        End Sub

        Sub getAttribute(line As String)
            Dim start As Long
            Dim endString As Long
            Dim char As String
            Dim position As Long
            position = 1
            start = 1
            Do While char <> """"
                char = Mid(line, position, 1)
                position = position + 1
            Loop
            endString = position - 2
            cList(cListCounter) = Mid(line, start, endString)
            cListCounter = cListCounter + 1

        End Sub
        Function getX(position As Integer) As String

            Dim line As String
            Dim info() As String
            line = data(position)
            info = Split(line, ",")
            getX = info(0) / 1000
        End Function
        Function getY(position As Integer) As String

            Dim line As String
            Dim info() As String
            line = data(position)
            info = Split(line, ",")
            getY = info(1) / 1000
        End Function
        Function getHeight(position As Integer) As String
            Dim line As String
            Dim info() As String
            line = data(position)
            info = Split(line, "=")
            getHeight = info(1) / 1000
        End Function
```

MACRO MAIN CODE

```
' ************************************************************************
' ************************************************************************
Dim swApp As Object
Dim Part As Object
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long
Dim swSketchMgr As SldWorks.SketchManager
Dim vSkLines As Variant
Dim swSketch As SldWorks.Sketch
Dim swFeat As SldWorks.Feature

Public strFileToOpen As String
Public data() As String
Public arraySize As Integer
```

```
15. Public cListCounter As Integer
16.
17.
18.
19.
20. Sub main()
21.
22. Set swApp = _
23. Application.SldWorks
24.
25. Set Part = swApp.ActiveDoc
26. 'read XML file (load Components)
27. 'grab list of components- xml
28. 'compare to sketches
29. 'display the ones not placed
30. 'check for plane selected
31.     ' yes - save it
32.     'no - ask for one
33. 'build selected components
34. 'allow to be placed
35.
36. Load AddComponent
37. AddComponent.Show (vbModeless)
38. 'Debug.Print list
39.
40. cListCounter = 0
41.
42. End Sub
```

APPENDIX-D

GENERATED G-CODE SNIPPET OF EMBEDDED CIRCUIT

```
;LAYER:2
G0 X32.667 Y98.500 Z0.800
;TYPE:FILL
G1 F2100 X101.766 Y29.401 E466.55511
G0 F9000 X103.264 Y31.493
G1 F2100 X90.327 Y18.555 E466.81511
G0 F9000 X88.833 Y17.585
G1 F2100 X20.853 Y85.566 E468.18127
G0 F9000 X22.826 Y88.542
G1 F2100 X91.810 Y19.559 E469.56758
G0 F9000 X85.704 Y15.764
G1 F2100 X19.033 Y82.436 E470.90744
G0 F9000 X17.446 Y79.318
G1 F2100 X42.501 Y104.373 E471.41095
G0 F9000 X41.930 Y104.086
G1 F2100 X107.353 Y38.663 E472.72571
G0 F9000 X105.682 Y35.385
G1 F2100 X38.652 Y102.415 E474.07277
G0 F9000 X35.584 Y100.533
G1 F2100 X103.799 Y32.318 E475.44364
G0 F9000 X99.552 Y26.666
G1 F2100 X29.932 Y96.286 E476.84275
G0 F9000 X27.419 Y93.849
G1 F2100 X97.116 Y24.151 E478.24342
G0 F9000 X94.545 Y21.773
G1 F2100 X25.041 Y91.277 E479.64019
G0 F9000 X17.379 Y79.140
G1 F2100 X82.409 Y14.111 E480.94705
G0 F9000 X80.014 Y13.192
G1 F2100 X108.628 Y41.806 E481.52208
G0 F9000 X108.775 Y42.191
G1 F2100 X45.458 Y105.508 E482.79452
G0 F9000 X49.194 Y106.722
G1 F2100 X109.988 Y45.928 E484.01626
G0 F9000 X110.789 Y48.917
G1 F2100 X72.902 Y11.030 E484.77765
G0 F9000 X70.958 Y10.711
G1 F2100 X13.979 Y67.691 E485.92273
G0 F9000 X13.651 Y65.623
G1 F2100 X56.195 Y108.168 E486.77772
G0 F9000 X57.455 Y108.361
G1 F2100 X111.627 Y54.189 E487.86638
G0 F9000 X111.657 Y54.735
G1 F2100 X67.086 Y10.163 E488.76210
G0 F9000 X66.584 Y10.137
G1 F2100 X13.404 Y63.316 E489.83082
G0 F9000 X13.243 Y60.266
G1 F2100 X61.553 Y108.576 E490.80167
G0 F9000 X62.157 Y108.608
G1 F2100 X111.874 Y58.891 E491.80080
G0 F9000 X111.863 Y59.891
G1 F2100 X61.929 Y9.957 E492.80429
G0 F9000 X61.808 Y9.963
G1 F2100 X13.230 Y58.540 E493.78052
G0 F9000 X13.391 Y55.464
```

```
G1 F2100 X66.354 Y108.427 E494.84489
G0 F9000 X67.339 Y108.376
G1 F2100 X111.642 Y64.073 E495.73522
G0 F9000 X111.600 Y64.577
G1 F2100 X57.243 Y10.220 E496.82759
G0 F9000 X56.481 Y10.340
G1 F2100 X13.607 Y53.214 E497.68920
G0 F9000 X13.946 Y51.069
G1 F2100 X70.750 Y107.873 E498.83076
G0 F9000 X73.230 Y107.434
G1 F2100 X110.701 Y69.964 E499.58378
G0 F9000 X110.923 Y68.850
G1 F2100 X52.969 Y10.896 E500.74844
G0 F9000 X50.350 Y11.520
G1 F2100 X14.788 Y47.083 E501.46312
G0 F9000 X14.813 Y46.987
G1 F2100 X74.832 Y107.005 E502.66927
G0 F9000 X78.657 Y105.881
G1 F2100 X15.938 Y43.161 E503.92970
G0 F9000 X17.296 Y39.625
G1 F2100 X42.893 Y14.028 E504.44411
G0 F9000 X41.771 Y14.548
G1 F2100 X107.273 Y80.049 E505.76045
G0 F9000 X105.602 Y83.328
G1 F2100 X38.492 Y16.218 E507.10911
G0 F9000 X35.439 Y18.115
G1 F2100 X103.705 Y86.381 E508.48101
G0 F9000 X102.819 Y87.745
G1 F2100 X91.011 Y99.553 E508.71831
G0 F9000 X91.676 Y99.101
G1 F2100 X22.718 Y30.143 E510.10411
G0 F9000 X24.933 Y27.408
G1 F2100 X94.411 Y96.886 E511.50037
G0 F9000 X96.995 Y94.521
G1 F2100 X27.298 Y24.823 E512.90103
G0 F9000 X29.800 Y22.375
G1 F2100 X99.444 Y92.020 E514.30063
G0 F9000 X101.660 Y89.285
G1 F2100 X32.535 Y20.161 E515.68978
G0 F9000 X20.757 Y33.132
G1 F2100 X88.687 Y101.061 E517.05492
G0 F9000 X85.545 Y102.869
G1 F2100 X18.950 Y36.274 E518.39323
G0 F9000 X17.311 Y39.585
      G1 F2100 X82.234 Y104.508 E519.69795
      G0 F9000 X80.403 Y105.211
      G1 F2100 X108.478 Y77.136 E520.26215
      G0 F9000 X108.709 Y76.536
      G1 F2100 X45.284 Y13.111 E521.53676
      G0 F9000 X49.005 Y11.882
      G1 F2100 X109.938 Y72.815 E522.76129
      G0 F9000 X110.956 Y49.910
      G1 F2100 X53.176 Y107.690 E523.92246
      G0 F9000 X50.020 Y106.942
      G1 F2100 X14.876 Y71.798 E524.62873
      G0 F9000 X14.865 Y71.755
      G1 F2100 X75.022 Y11.597 E525.83767
      G0 F9000 X78.831 Y12.738
      G1 F2100 X16.006 Y75.564 E527.10024
      G0 F9000 X15.491 Y74.566
      ;TYPE:WALL-INNER
```

```
117.        G1 F2100 X14.151 Y69.567 E527.17378
118.        G1 X13.340 Y64.441 E527.24753
119.        G1 X13.069 Y59.277 E527.32101
120.        G1 X13.340 Y54.112 E527.39451
121.        G1 X14.151 Y48.986 E527.46826
122.        G1 X15.491 Y43.988 E527.54179
123.        G1 X17.346 Y39.156 E527.61534
124.        G1 X19.699 Y34.538 E527.68899
125.        G1 X22.518 Y30.197 E527.76254
126.        G1 X25.778 Y26.171 E527.83615
127.        G1 X29.438 Y22.511 E527.90971
128.        G1 X33.464 Y19.251 E527.98332
129.        G1 X37.805 Y16.432 E528.05687
130.        G1 X42.423 Y14.079 E528.13052
131.        G1 X47.255 Y12.224 E528.20407
132.        G1 X52.253 Y10.884 E528.27761
133.        G1 X57.379 Y10.073 E528.35135
134.        G1 X62.544 Y9.802 E528.42485
135.        G1 X67.708 Y10.073 E528.49833
136.        G1 X72.834 Y10.884 E528.57208
137.        G1 X77.833 Y12.224 E528.64563
138.        G1 X82.665 Y14.079 E528.71918
139.        G1 X87.283 Y16.432 E528.79283
140.        G1 X91.624 Y19.251 E528.86638
141.        G1 X95.649 Y22.511 E528.93998
142.        G1 X99.310 Y26.171 E529.01355
143.        G1 X102.570 Y30.197 E529.08716
144.        G1 X105.389 Y34.538 E529.16071
145.        G1 X107.742 Y39.156 E529.23436
146.        G1 X109.597 Y43.988 E529.30791
147.        G1 X110.936 Y48.986 E529.38144
148.        G1 X111.748 Y54.112 E529.45519
149.        G1 X112.019 Y59.277 E529.52869
150.        G1 X111.748 Y64.441 E529.60217
151.        G1 X110.936 Y69.567 E529.67592
152.        G1 X109.597 Y74.566 E529.74946
153.        G1 X107.742 Y79.398 E529.82301
154.        G1 X105.389 Y84.016 E529.89666
155.        G1 X102.570 Y88.357 E529.97021
156.        G1 X99.310 Y92.382 E530.04382
157.        G1 X95.649 Y96.043 E530.11739
158.        G1 X91.624 Y99.303 E530.19099
159.        G1 X87.283 Y102.122 E530.26455
160.        G1 X82.665 Y104.475 E530.33820
161.        G1 X77.833 Y106.330 E530.41175
162.        G1 X72.834 Y107.669 E530.48529
163.        G1 X67.708 Y108.481 E530.55904
164.        G1 X62.544 Y108.752 E530.63252
165.        G1 X57.379 Y108.481 E530.70602
166.        G1 X52.253 Y107.669 E530.77977
167.        G1 X47.255 Y106.330 E530.85330
168.        G1 X42.423 Y104.475 E530.92685
169.        G1 X37.805 Y102.122 E531.00050
170.        G1 X33.464 Y99.303 E531.07405
171.        G1 X29.438 Y96.043 E531.14766
172.        G1 X25.778 Y92.382 E531.22123
173.        G1 X22.518 Y88.357 E531.29483
174.        G1 X19.699 Y84.016 E531.36838
175.        G1 X17.346 Y79.398 E531.44203
176.        G1 X15.491 Y74.566 E531.51558
177.        G0 F9000 X15.158 Y74.675
```

```
178.        ;TYPE:WALL-OUTER
179.        G1 F2100 X13.808 Y69.637 E531.58970
180.        G1 X12.993 Y64.486 E531.66381
181.        G1 X12.719 Y59.277 E531.73793
182.        G1 X12.993 Y54.067 E531.81207
183.        G1 X13.808 Y48.916 E531.88618
184.        G1 X15.158 Y43.879 E531.96028
185.        G1 X17.026 Y39.012 E532.03436
186.        G1 X19.394 Y34.366 E532.10846
187.        G1 X22.235 Y29.990 E532.18260
188.        G1 X25.514 Y25.941 E532.25664
189.        G1 X29.208 Y22.247 E532.33087
190.        G1 X33.257 Y18.968 E532.40491
191.        G1 X37.633 Y16.127 E532.47905
192.        G1 X42.279 Y13.759 E532.55315
193.        G1 X47.146 Y11.891 E532.62724
194.        G1 X52.183 Y10.541 E532.70134
195.        G1 X57.334 Y9.726 E532.77545
196.        G1 X62.544 Y9.452 E532.84958
197.        G1 X67.753 Y9.726 E532.92371
198.        G1 X72.904 Y10.541 E532.99782
199.        G1 X77.942 Y11.891 E533.07193
200.        G1 X82.809 Y13.759 E533.14601
201.        G1 X87.455 Y16.127 E533.22011
202.        G1 X91.831 Y18.968 E533.29425
203.        G1 X95.879 Y22.247 E533.36828
204.        G1 X99.574 Y25.941 E533.44253
205.        G1 X102.853 Y29.990 E533.51657
206.        G1 X105.694 Y34.366 E533.59071
207.        G1 X108.062 Y39.012 E533.66481
208.        G1 X109.930 Y43.879 E533.73889
209.        G1 X111.279 Y48.916 E533.81299
210.        G1 X112.095 Y54.067 E533.88710
211.        G1 X112.369 Y59.277 E533.96124
212.        G1 X112.095 Y64.486 E534.03536
213.        G1 X111.279 Y69.637 E534.10947
214.        G1 X109.930 Y74.675 E534.18358
215.        G1 X108.062 Y79.542 E534.25766
216.        G1 X105.694 Y84.188 E534.33176
217.        G1 X102.853 Y88.564 E534.40590
218.        G1 X99.574 Y92.612 E534.47993
219.        G1 X95.879 Y96.307 E534.55419
220.        G1 X91.831 Y99.586 E534.62821
221.        G1 X87.455 Y102.427 E534.70235
222.        G1 X82.809 Y104.795 E534.77646
223.        G1 X77.942 Y106.663 E534.85054
224.        G1 X72.904 Y108.012 E534.92465
225.        G1 X67.753 Y108.828 E534.99876
226.        G1 X62.544 Y109.102 E535.07288
227.        G1 X57.334 Y108.828 E535.14702
228.        G1 X52.183 Y108.012 E535.22113
229.        G1 X47.146 Y106.663 E535.29523
230.        G1 X42.279 Y104.795 E535.36931
231.        G1 X37.633 Y102.427 E535.44341
232.        G1 X33.257 Y99.586 E535.51755
233.        G1 X29.208 Y96.307 E535.59159
234.        G1 X25.514 Y92.612 E535.66584
235.        G1 X22.235 Y88.564 E535.73986
236.        G1 X19.394 Y84.188 E535.81400
237.        G1 X17.026 Y79.542 E535.88811
238.        G1 X15.158 Y74.675 E535.96219
```

```
239.        G0 F9000 X15.985 Y74.380
240.        G0 X25.293 Y116.864
241.        ;TYPE:FILL
242.        G1 F2100 X25.998 Y117.568 E535.97634
243.        G0 F9000 X25.998 Y120.019
244.        G1 F2100 X25.293 Y120.723 E535.99050
245.        G0 F9000 X25.293 Y121.813
246.        G1 F2100 X25.998 Y122.518 E536.00467
247.        G0 F9000 X25.998 Y124.969
248.        G1 F2100 X25.622 Y125.345 E536.01223
249.        G0 F9000 X23.875 Y125.345
250.        G1 F2100 X21.171 Y122.641 E536.06657
251.        G0 F9000 X23.376 Y122.641
252.        G1 F2100 X20.672 Y125.345 E536.12091
253.        G0 F9000 X18.925 Y125.345
254.        G1 F2100 X17.293 Y123.713 E536.15370
255.        G0 F9000 X17.294 Y123.773
256.        G1 F2100 X18.426 Y122.641 E536.17645
257.        G0 F9000 X18.155 Y122.485
258.        G0 X17.997 Y119.467
259.        G1 F2100 X17.293 Y118.763 E536.19060
260.        G0 F9000 X17.294 Y118.823
261.        G1 F2100 X17.997 Y118.120 E536.20473
262.        G0 F9000 X18.121 Y116.519
263.        ;TYPE:WALL-INNER
264.        G1 F2100 X18.121 Y122.519 E536.28999
265.        G1 X25.171 Y122.519 E536.39017
266.        G1 X25.171 Y116.519 E536.47544
267.        G1 X26.121 Y116.519 E536.48893
268.        G1 X26.121 Y125.469 E536.61612
269.        G1 X17.171 Y125.469 E536.74330
270.        G1 X17.171 Y116.519 E536.87048
271.        G1 X18.121 Y116.519 E536.88398
272.        G0 F9000 X18.471 Y116.169
273.        ;TYPE:WALL-OUTER
274.        G1 F2100 X18.471 Y122.169 E536.96924
275.        G1 X24.821 Y122.169 E537.05948
276.        G1 X24.821 Y116.169 E537.14474
277.        G1 X26.471 Y116.169 E537.16819
278.        G1 X26.471 Y125.819 E537.30531
279.        G1 X16.821 Y125.819 E537.44244
280.        G1 X16.821 Y116.169 E537.57957
281.        G1 X18.471 Y116.169 E537.60302
282.        G0 F9000 X17.596 Y117.044
283.        ;TYPE:CUSTOM CIRCUIT
284.        (*** LAYER: 0 ***)
285.        T1 M6
286.        S6000
287.
288.        (* SHAPE Nr: 0 *)
289.        G0 X67.346 Y 24.7525
290.        M3 M8
291.        G0 Z   3.000
292.        F150
293.        G1 Z  -1.500
294.        F400
295.        G1 X68.249 Y 45.851
296.        G1 X70.042 Y 94.248
297.        G1 X70.452 Y 120.013
298.        G1 X71.89 Y 124.549
299.        G1 X74.69 Y 128.398
```

```
300.        G1 X78.536 Y 131.206
301.        G1 X85.419 Y 132.989
302.        G1 X92.416 Y 131.701
303.        G1 X98.285 Y 127.673
304.        G1 X102.095 Y 121.656
305.        G1 X103.319 Y 114.637
306.        G1 X101.854 Y 107.661
307.        G1 X99.142 Y 103.038
308.        G1 X95.291 Y 99.309
309.        G1 X90.596 Y 96.72
310.        G1 X85.392 Y 95.427
311.        G1 X78.267 Y 95.814
312.        G1 X71.658 Y 98.51
313.        G1 X66.257 Y 103.177
314.        G1 X62.271 Y 110.141
315.        G1 X60.914 Y 118.05
316.        G1 X62.293 Y 125.957
317.        G1 X66.196 Y 132.974
318.        G1 X72.155 Y 138.357
319.        G1 X79.512 Y 141.578
320.        G1 X87.507 Y 142.352
321.        G1 X95.358 Y 140.649
322.        G1 X102.337 Y 136.666
323.        G1 X107.829 Y 130.801
324.        G1 X111.38 Y 123.591
325.        G1 X112.72 Y 115.666
326.        G1 X111.77 Y 107.683
327.        G1 X108.64 Y 100.278
328.        G1 X103.6 Y 94.014
329.        G1 X97.055 Y 89.344
330.        G1 X88.035 Y 86.303
331.        G1 X78.515 Y 86.207
332.        G1 X69.419 Y 89.017
333.        G1 X61.588 Y 94.436
334.        G1 X55.716 Y 101.934
335.        G1 X52.295 Y 110.823
336.        G1 X51.586 Y 120.322
337.        G1 X53.614 Y 129.63
338.        G1 X57.559 Y 137.13
339.        G1 X63.243 Y 143.417
340.        G1 X70.296 Y 148.117
341.        G1 X78.279 Y 150.962
342.        G1 X86.714 Y 151.803
343.        G1 X95.107 Y 150.613
344.        G1 X102.983 Y 147.477
345.        G1 X109.91 Y 142.59
346.        G1 X115.519 Y 136.233
347.        G1 X118.684 Y 130.712
348.        G1 X120.858 Y 124.73
349.        G1 X121.981 Y 118.464
350.        G1 X122.031 Y 112.1
351.        G1 X118.725 Y 22.849
352.        F150
353.        G1 Z   3.000
354.        G0 Z  15.000
355.        M9 M5
356.
357.        G0 X84.469 Y 116.4925
358.        M2 (Program end)
359.        ;LAYER:3
360.        G0 X32.667 Y98.500 Z1.050
```

Vita

Efrain Aguilera Jr born in Illinois is the son of Mr. Efrain Aguilera and Maria Elena Aguilera. Efrain attended Coronado High School and received a High School Diploma in 2011. Immediately after, he started at the University Of Texas at El Paso, becoming an electrical engineering student while working as a research assistant in a 3D printing lab. Efrain received a Bachelor's of Science in Electrical Engineering in the spring 2015. During his undergraduate studies Efrain presented in Solid Free Form Fabrication Symposium and the Second Printed Electronics Workshop for the DoD Community. Efrain is also credited for creating one of the first 3D printed electromechanical parts, a 3 phase motor.

Contact Information: eaguilera5@miners.utep.edu

This thesis was typed by Efrain Aguilera Jr