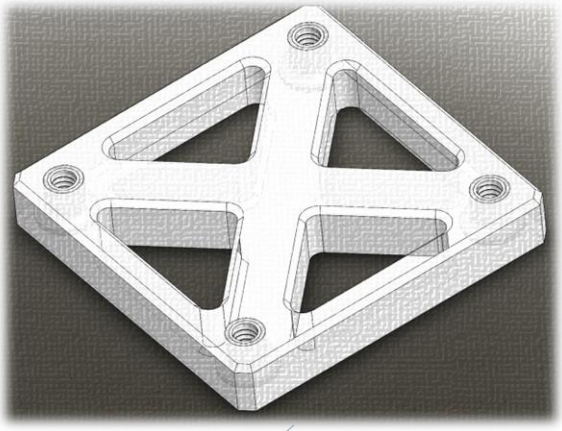
TEAM ADD_IN

05.05.2016



Final Design Review

TEAM F

- Nikhil Baheti
- Daniel Berman
- Ihsane Debbache
- Astha Prasad

Table of Contents

1 Project Description

The ADD_IN team aims to develop a 3D printer technology (named 4DOF) capable of producing printed parts which incorporate commercial off the shelf (COTS) components. These COTS items can include components such as threaded inserts, structural reinforcement (stiffeners), and electronics. Incorporating these components enables inclusion of geometrically precise features (such as threads), production of stronger parts (via the use of stiffeners), and reduced overall production time (since post-printing operations such as tapping and gluing can be eliminated). It will also open up new applications for 3D printing by making it possible to build complex parts that are hard or impossible to make otherwise.

The parts to be incorporated will be generally cylindrical or rectangular prism in shape, can optionally extend up to 1" above the top layer of the part, and must have sides which are orthogonal to the print surface. The 3D print material will be precisely deposited along the perimeter of the COTS item, thus encapsulating it within the 3D printed part.

Within the scope of this project, the installation of COTS parts during the printing process will be performed by a human operator. When the 3D printer reaches the print layer on which a part is to be installed, it will 'pause', move to a safe and accessible configuration for COTS item insertion, and await a command from the user that the part has been successfully installed before resuming printing. The system will be able to produce printed parts incorporating multiple COTS items at varying locations and orientations, provided that the orientations and clearances between the parts do violate certain geometric restrictions.

2 Use Case

An RC plane engineering company is in a race to develop a new fixed-wing airplane. The general configuration of the airplane has been determined but continued iteration on the structure and aerodynamic surfaces is needed. Up until this point the airplane has been prototyped using individually cut pieces of foam, balsa wood, and aluminum. Even with the aid of a laser cutter and CNC hot-wire foam cutter this has been a time consuming process since many of the parts are made from vastly different materials and are difficult to join together. Because of this the engineering team is often using glues and composites to hold the parts together, which makes it near impossible to replace or modify parts thus making design iteration difficult.

As with their other fixed wing airplanes, and as is an industry norm, the company eventually aims to mass produce the design from aluminum and polystyrene. Aluminum provides the structural rigidity necessary to form a solid airframe, especially for enduring the high stresses present at the motor mounts. Polystyrene is a light-weight plastic which can be easily molded into the complex shapes necessary to produce the aerodynamic surfaces of the aircraft. Furthermore, polystyrene can be injection molded to contain thread inserts to provide attachment points to the airframe and other internal

Figure 1: Injection molded part with threaded insert. (http//precision.bc.ca/photos/)

hardware. This however, is only cost effective for large production runs.

The design team has previously tried using 3D printers to more rapidly produce and test structural components. While helpful in some instances, the team has generally found that directly replacing their wood-polystyrene-composite prototypes with 3D printed parts often leads to failure at the mechanical interfaces between parts. Efforts to work around this problem have been made, but usually involve designing a special 'thickened' version of the part for 3D printing, which must then be redesigned when transitioning to large volume manufacturing using polystyrene. This extra design step uses up valuable team member time and changes the weight, balance, and aero elastic properties of the aircraft. The team however has remarked that 3D printing's ability to easily incorporate more complex features, such as wire runs, electronics mounts, and access ports into their project is a great benefit.

Figure 2: Example of failed 3D printed motor mount and a less desirable 'thickened' replacement necessary to support the loads. (http://www.wattflyer.com)

Recently, as part of an experiment to improve their design process, the aircraft design team purchased an ADD_IN 4DOF 3D printer. This printer contains an extra mechanical degree of freedom and special slicing software to enable 3D printing of complex parts that include embedded COTS items. The cost of the printer was comparable to other FDM 3D printers, and thus was not a significant burden to the small company. Since the printer looked and operated in nearly the same manner as their other 3D printers, the employees were immediately able to start using the printer without any additional training.

The first project the 4DOF printer was applied to was development of the aircraft's motor mount. The motor mount is a particularly challenging component since it must be strong enough to withstand large forces and torques imparted by the motor, propeller, and occasional crash landings, but also be light weight since it is the most forward part of the aircraft where weight is a premium. In these regards 3D printed parts had previously failed due to their poor material properties, especially with the motor mounting bolts loosening, stripping and pulling out from the plastic part. Using the ADD_IN printer the team was able to produce a 3D printed part directly from the airplane's CAD design, and include threaded inserts which could securely attach the motor and distribute the forces within the motor mount thus preventing material failure. The new motor mount was produced in half the time and with fewer resources than the previous aluminum-balsa wood and glue version, and thus enabled the team to conduct more frequent tests and design iterations.

Figure 4: Time intensive aluminum and balsa motor mount. (http://www.rcgroups.com)

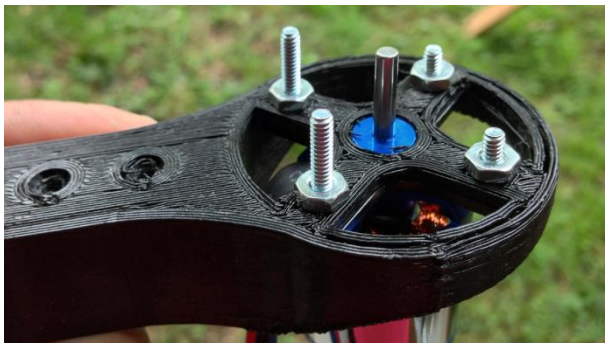


Figure 3: Example 3D printed motor mount with stripped threads (https://backyardrc.wordpress.com/category/quadcopter/)

Figure 5: Motor mount made with ADD_IN printer (http://www.anzel360.com/category/rc/fixed-wing/)

In another instance, the engineering team decided to experiment with using the 4DOF printer to produce a wing rib. This particular wing rib had been causing difficulty since it established both the aerodynamic profile of the wing, but also was the structural mount for the mechanism that actuated the ailerons. Because of this it needed to be particularly rigid, and provide mounting for numerous bearings, sensors, and the aileron motor, and have a precise complex outer geometry to define the airfoil shape of the wing. All of these requirements had made prototyping in balsa-wood particularly difficult, and often the team had to pay large sums to have the part CNC machined.

Immediately, the 4DOF printer was able to overcome these issues. Incorporation of screw inserts, aluminum stiffeners, and a special aluminum socket for connection with the wing spar provided the structural rigidity needed. Because 4DOF printers can produce complex 3-dimensional parts, the bearing and motor mounts needed were not only incorporated into the wing rib, but actually replaced with integrated features in the rib which directly held the bearings and motor and reduced the overall part count. Finally, since the 4DOF printer can print around complex shapes, even the sensors used in the aileron actuation mechanism could be incorporated directly into the wing rib, thus greatly simplifying the whole assembly. Again, producing the

wing rib required only a fraction of the time as needed for former methods, and the result achieved was much higher than previously achievable. Rapid design iteration perfected the part, and helped to significantly reduce the time-to-market of the RC airplane engineering company's newest airplane.

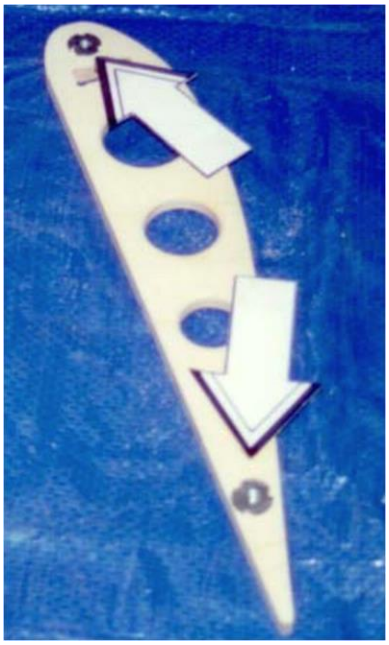


Figure 6: Plastic wing rib with added screw inserts. 4DOF printed ribs can have both more complex geometries and integrated screw inserts (https://sidewalkfliersag.wordpress.com/)

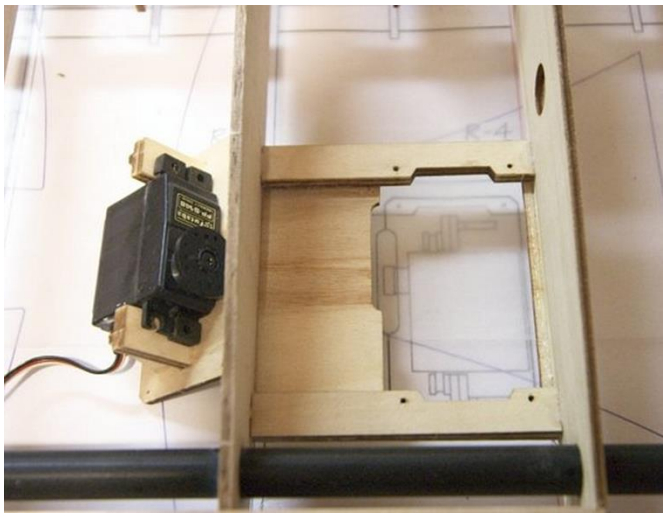


Figure 7: Balsa wood wing rib with complex aileron-servo mount

The case of the fixed-wing drone manufacturer is not unique. Thousands of companies today extensively rely on 3D printing to rapidly prototype parts and iterate designs. Improving the efficiency of this process even a small amount can have multiplicative effects on the quality of a resulting product. Strong, lightweight 3D printed parts and parts with integrated sensors and electronics represent a significant new advance in 3D printing technology that will help to provide rapid prototyping, mass customization, and lower cost parts to a variety of both businesses and consumers.

3 System Level Requirements

The system is divided into two primary focuses, namely the 'Software' and 'Hardware'. System requirements are mapped to either of these sections. Requirements that have been modified since the CDR are indicated in orange. Within Software and Hardware, the requirements are grouped under the categories listed in Table 1.

Table 1 : Requirements Labeling Legend

ID	Category
M	Mandatory (Threshold requirement)
D	Desirable (Stretch Goal)
F	Functional
N	Non Functional
P	Performance

3.1 Software Requirements

The Software System shall –

Table 2: Software Requirements

ID	Requirement	Subsystem
	Software Mandatory Functional Requirements	
M.F.1	**- Receive 3D files** The execution shall begin once the 3D file of the assembly (part to be printed + COTS item(s)) has been fed to the Slicer program.	**Slicer**
M.F.2	**- Prompt user for insertion layer** Slicer shall request the user to input the decided insertion layer. Upon reaching this layer, the printer will pause and move to a safe position and allow the COTS item to be inserted.	**Slicer**
M.F.3	**- Create collision free path** With knowledge of the dimensions and position of the COTS item, the Slicer program shall create a collision free path around the same.	**Slicer**
M.F.4	**- Generate 4DOF G-Code** The G-Code is comprised of movement commands for each of the 4 DOFs The printer interface software will load this G-Code before printing.	**Slicer**
	Software Mandatory Nonfunctional Requirements	
M.N.1	**- Work with standard 3D print files** The slicing program shall be compatible with various standard 3D files Examples include **.stl, .obj** and **.amf**	**Slicer**
M.N.2	**- Slicing, Insertion Layer Selection, and Path Planning to implemented in single software application** The slicing program shall be integrated to a single software application to allow the user to use it easily.	**Slicer**
	Software Desirable Functional Requirement	
D.F.1	**- Assign insertion layer**	

	A desirable feature of the slicing program would be the capability to assign insertion layers autonomously based on the dimensions and orientation of the COTS item. This would greatly reduce the program's dependence on humans.	**Slicer**

	Software Desirable Non Functional	
D.N.1	**- Easy-to-use interface** Since the system requires user interaction, it is necessary to have an intuitive graphical user interface.	**Slicer**

	Software Desirable Performance Requirements	
~~**D.P.1**~~	~~**- Print time not to increase more than 5% compared to Makergear M2**~~ ~~The total print time for the part should not increase more than 5% when compared to the current print time. This does not include time to add the COTs part.~~ REQUIREMET REMOVED	~~**Slicer**~~
D.P.2	**- Print between multiple COTS parts not less than 2 inches apart** The path planning algorithm must be able to path plan for multiple COTs part to be added	**Slicer**

3.2 Hardware Requirements

The Hardware System shall –

Table 3: Hardware Requirements

	Hardware Mandatory Functional Requirements	
M.F.1	**- Print layers of material** The primary function of the 3D printer, upon which our system is developed, must be upheld post alterations to the manipulator and other system components.	**Nozzle**
M.F.2	**- Print locating features for COTS items** Accurate positioning of an item on a partial print is achieved by printing locating features tailored to the base dimensions of the said item.	**Firmware**
M.F.3	**- Go to safe configuration during insertion** Since the addition of the COTS part involves a human placing the part on a partial print, measures shall be taken to ensure that the hot nozzle is out of the way and that the partially printed part is easily accessible. The above shall include: i. Moving the nozzle to safe location (away from printed part) ii. Fully lowering the bed to the insertion position	**Firmware**
M.F.4	**- Enclose COTS item with print material** The printer shall deposit print material close to the added part such that there is no clearance room around the piece for it to move, allowing for a snug fit.	**Rotation Stage**
M.F.5	**- Avoid Collisions** Since the added COTS part will be protruding above the top surface of the part, the movement of the printer nozzle will be controlled to avoid collisions with it.	**Rotation Stage**
M.F.6	**- Rotate nozzle Using G-code** The nozzle must be rotated to avoid the COTS item using trajectories executed by firmware (waypoints will be generated by software)	**Firmware**

M.F.7	**- Avoid Kinks in Filament** The rotary stage must not kink the filament while rotating.	**Rotation Stage**

Hardware Mandatory Non Functional Requirements		
M.N.1	**- Provide user feedback during printing** The user will be provided with a real time display of the progression of the print as the material gets deposited along the coordinates provided by the G-Code.	**LCD screen**
M.N.2	**- Maintain accurate temperature control** The slip ring adds noise to the analog data. Thus there may need to be a system that makes the data through the slip ring immune to noise.	**PCB**

Hardware Desirable Functional Requirements		
D.F.1	**- Print between parts** The nozzle should be capable of maneuvering between two added COTS items. The items shall not be more than **2 inches apart.**	**Nozzle**
D.F.2	**- Print close to the COTS item** The printer should print within **0.1 mm** of the COTS part	**Rotary Stage**

Hardware Desirable Non Functional Requirements		
D.N.1	**- Maintain print speed** The 4DOF printer should not have maximum axis velocities lower than that of the unmodified printer.	**Rotary Stage**

The performance requirements are as follows:

Hardware Mandatory Performance Requirements		
M.P.1	**- Incorporate COTS parts that are orthogonal to print surface** The hardware must be able to print against parts that are orthogonal to the print plane. Examples: Cylindrical and rectangular prism shapes	**Nozzle**
M.P.2	**- Incorporate COTS parts that have a maximum height of one inch above the print plane** The COTS part must have a maximum height of 1 inch because beyond that we need to move along negative z to avoid collision	**Nozzle**
M.P.3	**- Print volume of 3x3x3 inches** Since the nozzle is being modified, this should not reduce the print volume of the printer significantly	**Nozzle**
M.P.4	**- Be able to infinitely rotate nozzle** The wires moving from through the rotary joint should not hinder the rotation and thus the rotation must be done infinitely	**Rotary Stage**

Hardware Desired Performance Requirements		
1D.P.1	**- Position nozzle within 0.1mm of COTS part** The is the general accuracy of a 3D printer and thus the printer must be able to print within 0.1mm of the COTS part	**Rotary Stage**

4 Functional Architecture

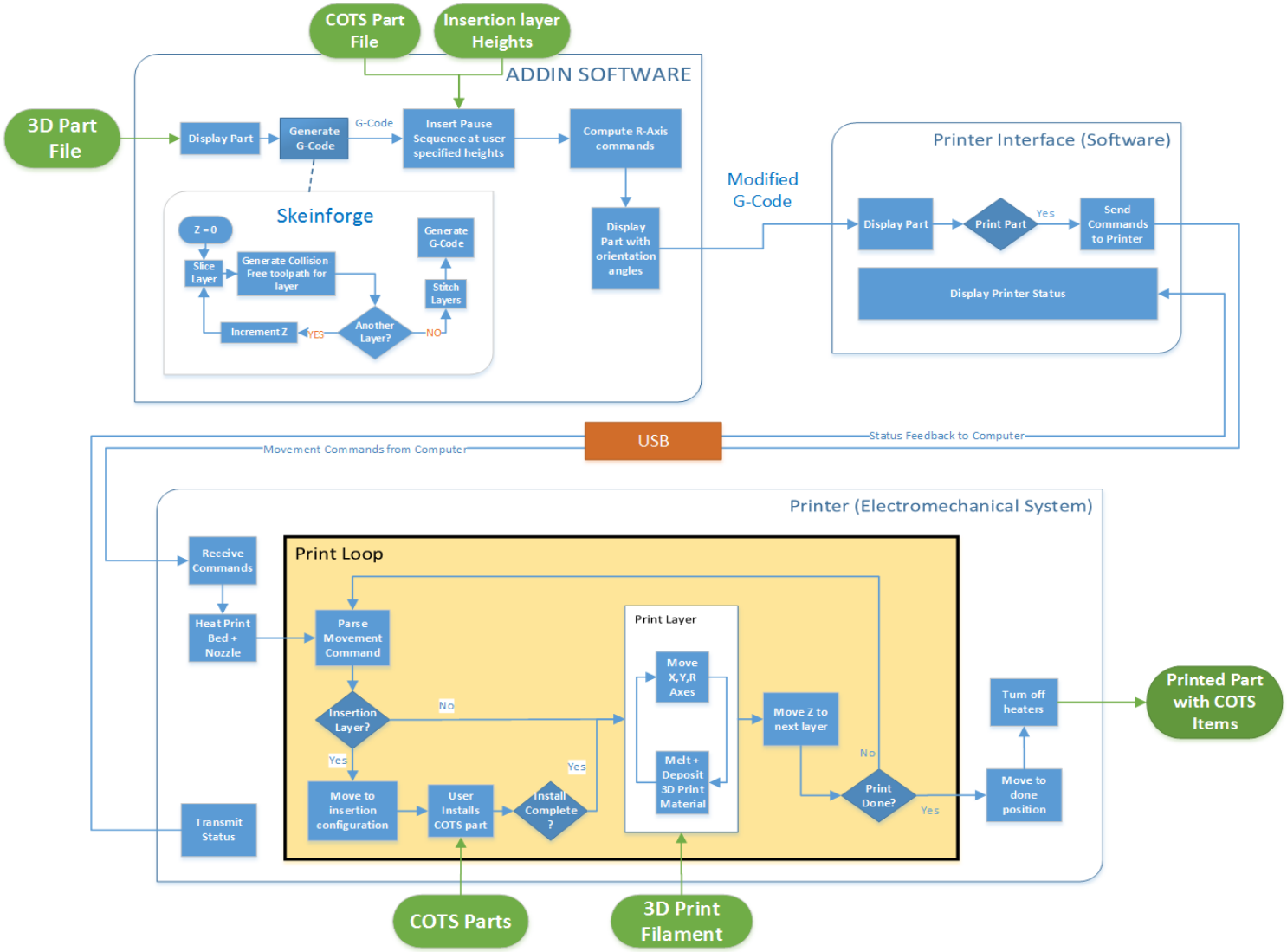


Figure 8: 4DOF functional architecture

4.1 Slicing Software

This subsystem is responsible for generating the G-Code that is fed to the printer. G-Code consists of a list of coordinates that define a path the printer nozzle must navigate while depositing the print material. The following is an example of a G-Code command. It includes a positioning command 'G00' followed by coordinate locations for the XYZ axes.

```
G0 X0 Y0 Z0.25
```

The subsystem was developed into an easy to use GUI. The user is allowed to select the STL file of the part to be printed. This part is then sliced to generate G-Code for the three existent XYZ axes by invoking a slicing program called 'Skeinforge' from command line. The user can vary the settings for the print, such as the layer height, infill pattern, extrusion rate, etc. Once the basic settings have been decided and G-Code generated, the user can select the STL file of the part to be inserted inside the printed body and position it in place by entering its XYZ position and orientation. After locking the COTS part in its correct location, the user must enter the insertion height at which the printer will pause to allow the COTS insertion. Based on the position of the inserted part, the obtained G-Code is modified to include the R-axis commands, which are nothing but the angle by which the 4^{th} stepper motor should rotate during the execution of each G-Code. The R-Axis movements ensure that the body of the nozzle and heat block do not collide with the COTS item.

This subsystem finally generates G-Code complete with commands to drive all the motors. This G-Code is then loaded into the printer interface to be fed to the printer line by line.

4.2 Printer Interface Software

The printer interface loads the G-Code generated by the slicing software and displays the 3D sliced model on the computer screen. It communicates with the microcontroller mounted on the printer over USB. The microcontroller receives instructions to initiate the print when the user hits 'Print'. While printing the interface streams G-Code commands to the printer, and the status is transmitted back from the printer to the software. The print status is displayed on the screen, giving the user real time feedback on the print progress.

4.3 Printer (Electromechanical System)

The printer is an electromechanical system that is responsible for physically producing the required 3D assembly. A soon as the microcontroller on board receives commands to initiate the print; it first begins to heat the bed and nozzle head to temperatures defined in the G-Code. The printer then begins printing the base structure layer by layer. If and when it encounters an insertion layer, the hot nozzle moves to the safe configuration and alerts the used to place the COTS item in its feature. The user then inserts the item and hits 'resume' which causes the nozzle to go back to its printing temperature and continue printing. Aided by the bent nozzle, extra degree of freedom and computed R-axis commands, the nozzle is able to smoothly enclose the added part in print material. Once the print had been completed, the heaters are turned off and the bed is rolled out to display the finished 3D part.

5 Cyberphysical Architecture

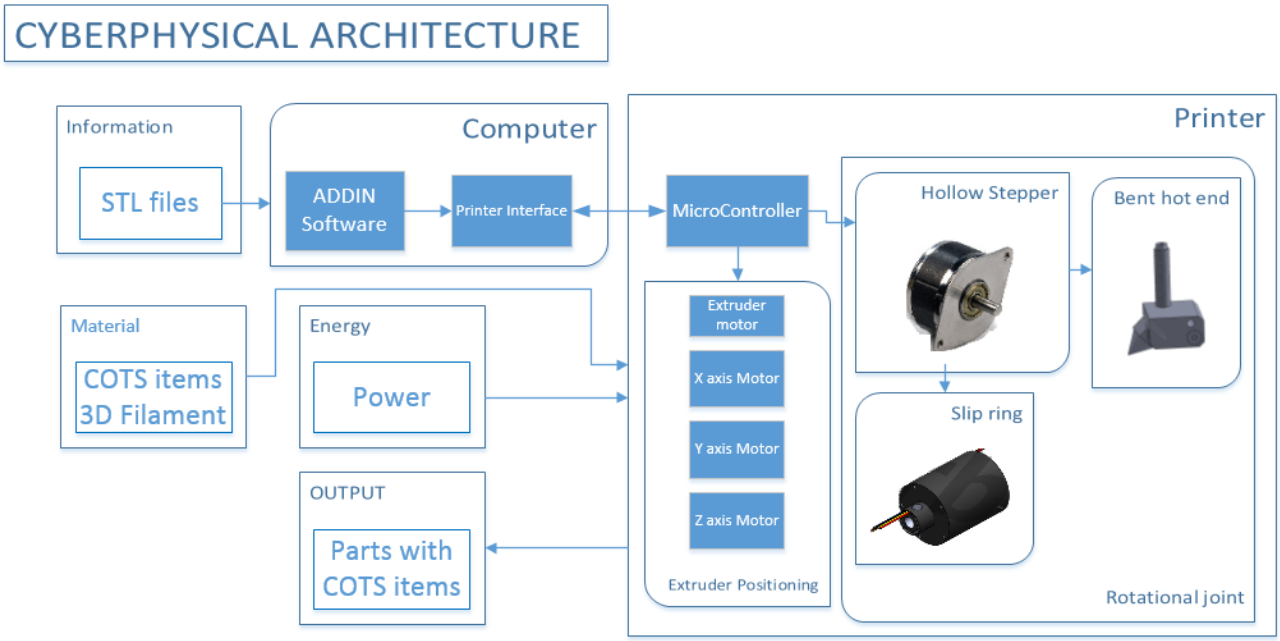


Figure 9: Cyberphysical Architecture

The cyberphysical architecture of the 4DOF system, shown in Figure 9, is inspired by the functional architecture and standard 3D printers. It comprises a software component, namely the ADDIN Software, Skeinforge Slicer and the printer interface, and a hardware component which is the 4DOF printer.

The 4DOF system is based on the Makergear M2 commercial 3D printer that was provided by our sponsor and already contains many of the necessary system components. The remaining portions of the cyberphysical architecture which are being developed by the ADD_IN team are:

1. **ADDIN software**: Our custom made software, that uses the Skeinforge Slicer to generate a standard 3DoF G-code and then computes and adds the 4th DoF.
2. T**he printer controller**: The Arduino based RAMBO Board for which we modified the Firmware to enable control of all degrees of freedom.
3. **R axis joint**: To avoid collisions while printing around the perimeter of COTS items the extruder nozzle must be infinitely rotatable. To enable infinite rotation a slip ring transfers heater power and temperature signals across the joint.
4. **Angled Nozzle**: A custom nozzle capable of printing along the surface of the COTS item

These components are explained in more details in the subsystems section of this report.

5.1 Description

Figure 10 provides a graphical illustration of the 4DOF printer. The components on the left describe how the nozzle and rotation joint will be physically mounted on the 3D printer.

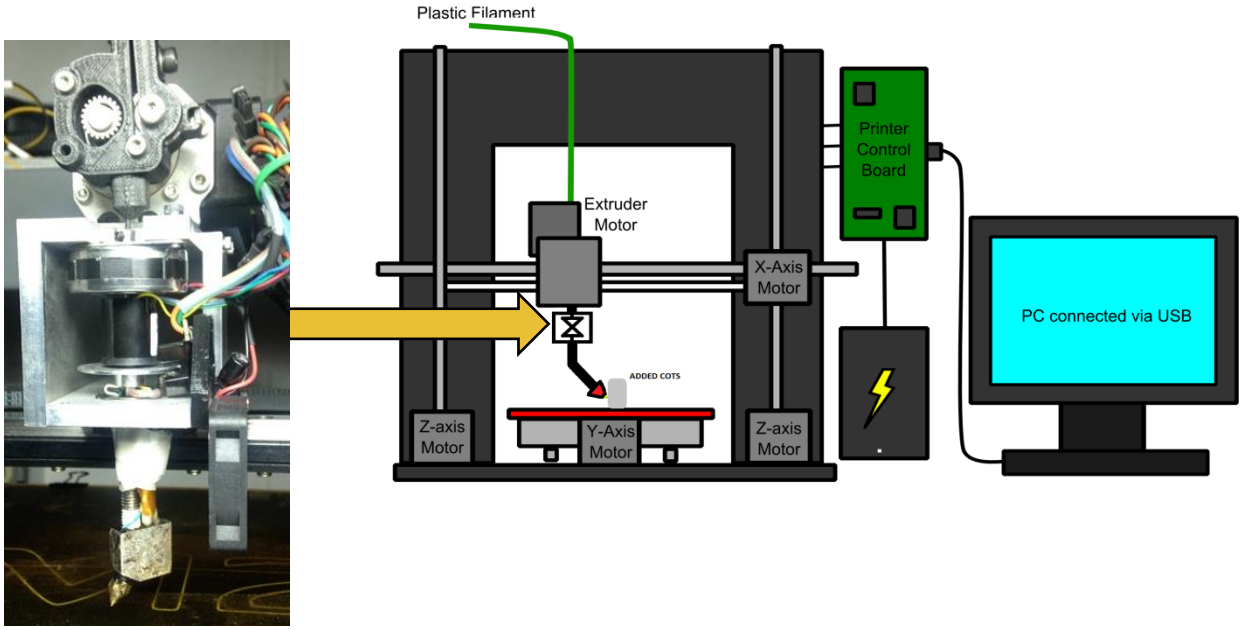


Figure 10: Graphical Depiction of 4DOF cyberphysical architecture

Many of the components in the cyberphysical architecture are from the original unmodified 3D printer. The main modification is highlighted in Figure 10 where it is shown how the hollow shaft stepper motor and the slip ring are mounted on a metallic mount that was machined in-house. The new mount provides support for the fan, encodes wheel and optical-sensor, and also comprises a bearing to reduce the flexibility of the nozzle.

6 System and Subsystems Description

6.1 Nozzle Subsystem

6.1.1 Description

The original nozzle on the Makergear M2 can only print parallel to the print bed. To print along the surface of COTS items a custom nozzle which can extrude filament at an angle is required. The nozzle subsystem describes the entire assembly which is required to melt and deposit filament including a mounting interface, heater, heat block, thermal insulator, thermistor, and small diameter brass nozzle.

6.1.2 Status

A custom nozzle was designed and manufactured to meet the system requirements. Multiple iterations and adjustments were performed to achieve a reliable nozzle with correct thermal and mechanical properties. Operation of the nozzle was demonstrated throughout the spring semester, and especially the SVE. A representation of the original and modified nozzle system is shown in Figure 11.

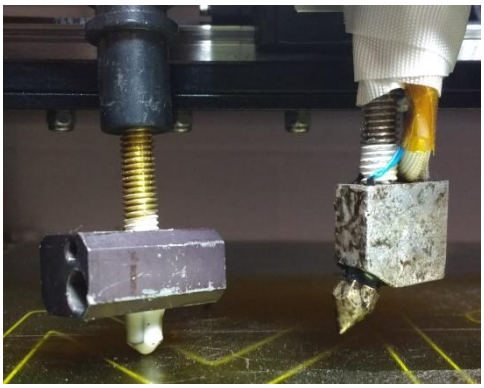


Figure 111: Changes from original 3D printer nozzle to custom 4DOF nozzle

The final iteration of the nozzle features a 30 degree bend, thus giving it the clearance to print on surfaces that are orthogonal to the print bed. Figure 12**Error! Reference source not found.** below shows the actual nozzle that was machined as per the nozzle design. This proved to be the most reliable one, the 30 degree angle and smaller aperture angle enables the nozzle to have a larger angle with respect to the print bed and not smudge the filament. The thinner heat block enables easier collision free planning with cots items.

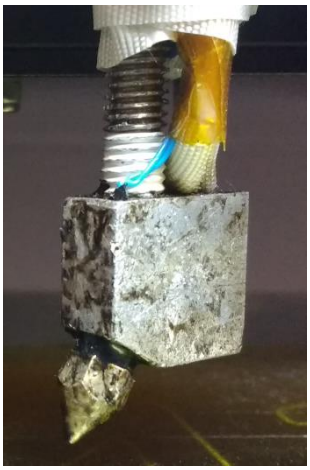


Figure 12: Custom 4DOF nozzle

6.1.3 Analysis

The nozzle was tested extensively to determine the correct operating parameters (temperature, print speed, nozzle diameter, nozzle angle, fan setting) that produces the best print quality. After many issues, especially jamming and smudging, the final iteration produces very reliable printing; the jams were resolved thanks to better heat transfer control, using a fan, insulator, and heat sinks. And the filament smudging is mitigated by reducing the nozzle aperture angle and the heat blocks bend angle.

6.1.4 Strengths and Weaknesses

The strengths of the nozzle subsystem include:

1. Consistent filament extrusion along preferred directions
2. Good temperature regulation (with insulation)

3. Modularity (Identical mounting interface as the original nozzle)

The weaknesses include:

1. Limited to convex parts that have 90° or less with respect to the bed. That is a design limitation since our angle is fixed.

6.2 Rotary Joint Subsystem

6.2.1 Description

The rotational joint subsystem is the mechanical portion of our system that provides an additional degree of freedom to the 3D printer necessary to avoid collisions with COTS items. The components of this subsystem are the stepper motor, the slip ring and the mount. Figure 133 shows a picture of these components.

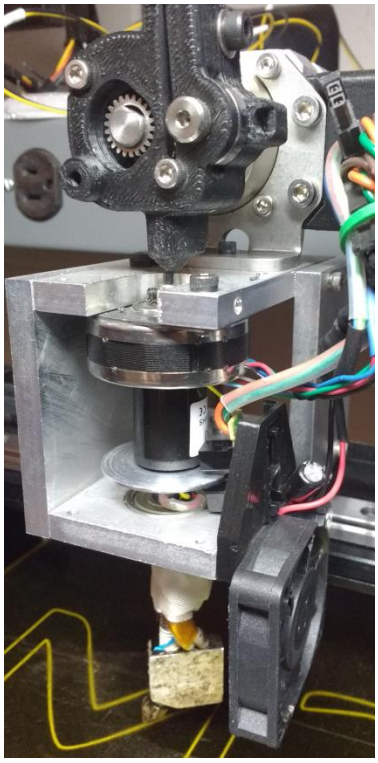


Figure 133: Rotational joint assembly.
From bottom left to top right: Hollow shaft stepper motor, slip ring, custom filament drive

The main design objectives for this subsystem were to minimize weight, flexibility and backlash, while maintaining a small form factor (especially the height as it will result in reduced print volume), and to satisfy our performance requirements for speed and precision. These are the main aspects that guided the selection of the components.

6.2.2 Analysis

Stepper motor

The motor we used is the 3709 stepper motor from Lin Engineering, custom made to have a hollow shaft. The specifications of importance for our system are as follows:
Repeatability: 1.5 % of a step = +-0.0135°
Holding Torque: 0.04 N-m
Weight: 80 grams
Steps per revolution: 400
Hollow shaft ID: 3 mm
Also the stepper is compatible with the A4988 stepper driver that is on our control board.

Slip ring

For the slip ring, we also needed a compact and lightweight component with at least 5 power lines, low added noise and hollow shaft. We selected the 504-0800 from Orbex, which satisfies these requirements. Figure 144 shows the stepper motor and slip ring.

Figure 144: Stepper motor (hollow shaft not shown) and slip ring

Mount

The mount was machined in-house to host all the rotary joint components, namely the stepper motor, slip ring, fan, encoder wheel and optical sensor and to connect with the hot end (bottom) as well as the filament drive(Top). A previously 3D printed version had some flexibility in the nozzle so we machined a metal version that also has a bearing to make it more rigid, the result is very rigid and reliable. Also the mount is directly mounted on the X-carriage, and the extruder motor lifted about 6cm higher, that was done by design in order to preserve the print volume, i.e. in this version the maximum print height is only reduced by 4cm.

6.2.3 SVE Evaluation

The rotary joint subsystem performed as expected for the SVE, homing was working reliably, and the nozzle can be rotated precisely and synchronously with the other axes.

6.2.4 Strengths and Weaknesses

Strengths:

The main strengths of the rotary joint subsystem are:

1. High Repeatability - 3 times the stated requirement.

2. Small form factor - Light weight
3. Simplicity - Having a concentric hollow shaft stepper motor enables a simple implementation, when compared to rotary stage, or standard stepper with gears.

Weaknesses:

1. No feedback: which is a weakness common to all 3D printers
2. Low stepper motor holding torque: Which causes the stepper to skip steps when encountering too much resistance

6.3 Microcontroller Subsystem

The microcontroller board that powers the Makergear M2 is the RAMBo 1.1B Controller Board. Rambo stands for RepRap Arduino-Mega Board. As shown in **Error! Reference source not found.** the Rambo board is an all-in-one electronics board that communicates with the printer interface software via USB and is used to control all components of the 3D printer.

6.3.1 Firmware

The board on our Makergear 2 is loaded with the Marlin Firmware, which is Arduino based and entirely open-source. The role of the Firmware is to interpret the G-code commands and control the printer's heaters and motions accordingly. The firmware has been extended to control a 4th degree of freedom which is the rotational axis. The rotational axis combined with above designed nozzle is used to avoid colliding with the COTs item and also improve the quality of print. To ensure that the print is as per the G-code the firmware incorporates the kinematics and inverse kinematics of the nozzle tip.

6.3.2 Modeling, analysis, and testing

Electrical Connections

The rotational axis is controlled using a stepper motor through the RAMBo board. A hollow stepper motor has been used as described above. To align the rotational axis and assign a homing to the R-axis we use an encoder wheel assembly with and optical encoder. The optical encoder is connected to the limit switch port. Figure15 shows the Rambo board extended connections used by ADD_IN.

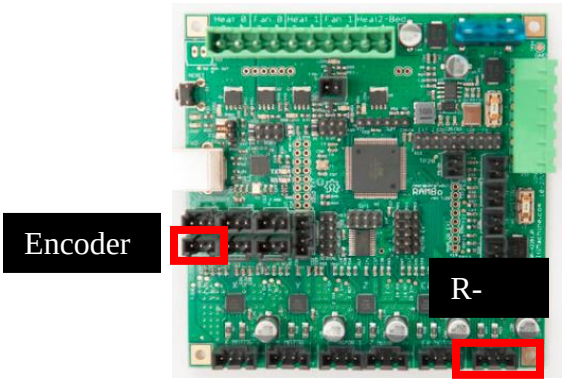


Figure 15: Extension ports used by ADD_IN

Kinematics and Inverse Kinematics

All the axes in the original firmware are moved synchronously with constant velocities. The x-axis and y-axis are moved in a straight line. This results in the extruder tip moving along a curve as shown in Figure 16. So, I worked out the inverse kinematics for the x and y location such that the extruder tip moves along the required straight line. The following are the equations for the inverse kinematics:

$$x_e = x - offset * \cos r$$
$$y_e = y - offset * \sin r$$

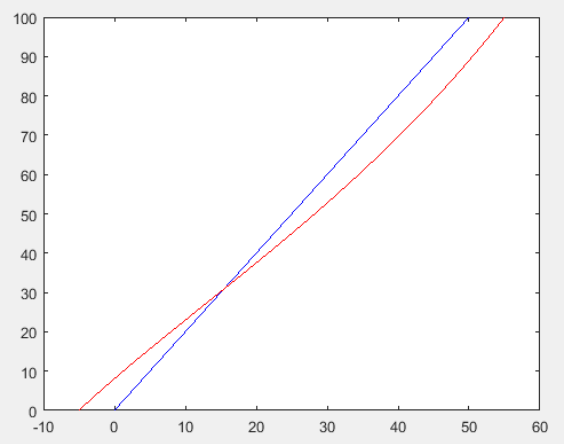


Figure 16: Red line is the motion of the extruder (x_e, y_e); blue line is the motion of the (x, y) stepper motors

Based on the above equations the velocities for the stepper motors are calculated so that the synchronous movement results in a straight line. The velocities of the stepper motor are defined as follows:

$$dx = dx_e - \cos(r + dr) + \cos(\mathrm{r});$$
$$dy = dy_e - \sin(r + dr) + \sin(r)\,;$$

where,
$dx_e \rightarrow$ *steps the nozzle tip move along* $X -$ *axis per timer interrupt*
$dx \rightarrow$ *steps the* $X -$ *axis stepper motor must move per timer interrupt*
$dy_e \rightarrow$ *steps the nozzle tip move along* $Y -$ *axis per timer interrupt*
$dy \rightarrow$ *steps the* $Y -$ *axis stepper motor must move per timer interrupt*
$dr \rightarrow$ *steps the* $R -$ *axis stepper motor must move per timer interrupt*
$r \rightarrow$ *current* $R -$ *axis angle*

To implement the above equations a look up table was implemented which consists of the trigonometric values. These were stored as integer values scaled to an appropriate magnitude so that all the computations could be performed quickly in the interrupt service routine. The implementation had drift issues which were solved by increasing the resolution of the computations. Figure17 shows the initial drift and the correct prints after solving the drift issues.

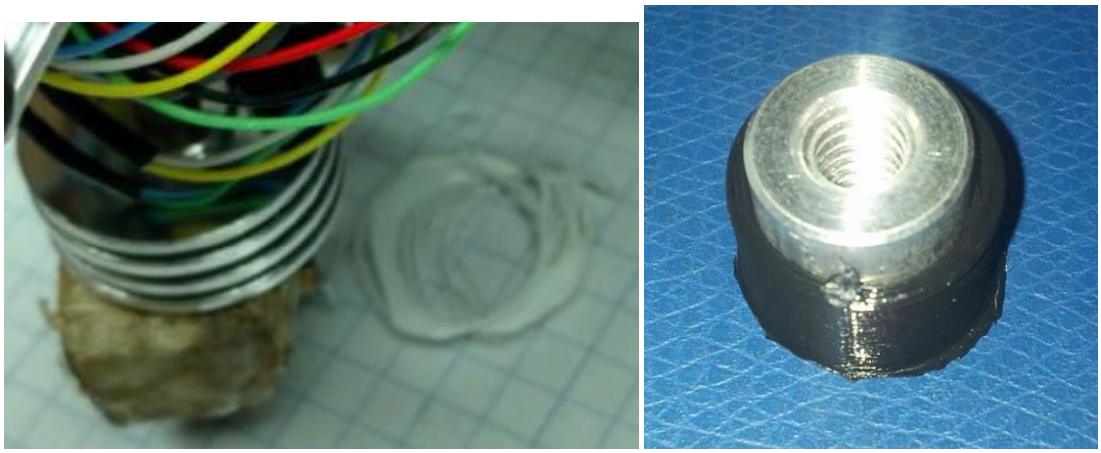


Figure 17: Comparison between print before and after solving drift

6.3.3 Strengths and Weaknesses

The strengths of the firmware are as follows:

- Motor Synchronization: The firmware can control up to 5 stepper motors synchronously
- Homing: The firmware is compatible with all axes homing. It has an homing offset to move the nozzle orientation based on the encoder wheel offset from the homing location
- Extend G-code feature: The firmware has been modified to ensure all G-codes are compatible with the new degree of freedom
- Offset settings: The firmware has an offset setting which can be used to define the nozzle offset for different printers

The weaknesses of the firmware subsystem are as follows:

- Speed: The computations are performed with long data types which take a lot of processing power. The usage of the ISR is now very critical and any further computations added in future will result in errors in the print process.

Reading G-code: If the print speed is increased by 10 folds then the firmware might fail reading a few G-code lines resulting in printing errors.

6.4 Software Subsystem

The software subsystem is developed into an easy to use GUI that encompasses all the necessary functions and allows the user to generate special G-Code at the click of a button. It also provides tools for visualization and accurate selection and placement of COTS parts.

6.4.1 Software Framework

The software framework was built to be robust by employing OOPS concepts that allowed smooth handling of data. The subsystem employs the use of three Classes: ADDIN, PrinterState and COTSItem

PrinterState

The class PrinterState binds all the state variables related to one G-Code command together. These include the XYZR positions of the motors, temperature of the extruder and bed, speed, extrusion rate, etc. With the help of this class, we are able to read G-Code line by line, translate all the state variables into an object of class PrinterState and hence create a table as shown

below. Once all the G-code has been read and converted to an array of states, we are free to process the data without having to deal with the .gcode file.

Table 4: G-Code represented in tabular format

Index	GCommand	X	Y	Z	R	E	F	Tbed	TNozzle
Obj 1	'G01'	2.434	1.876	1.5	30	10	1800	60	215
Obj 2	'G01'	4.234	4.764	1.5	30	15	1800	60	215
Obj 3	'G01'	5.876	8.764	1.5	30	20	1800	60	215
…									

ADDIN

The ADDIN class contains all the data and functions pertaining to one particular 3D part. Some of the functions it includes are:

- Read G-Code file into array of PrinterStates
- Compute R axis commands
- Plot the 3D part
- Write PrinterStates back into a .gcode file

COTSItem

An object of this class represents all the data related to one COTS item, which includes its dimensions, minimum and minimum z heights, etc.

6.4.2 Path Planning Algorithm

Description:

The path planning algorithm essentially computes the angles that the stepper must rotate by during the execution of each G-Code command in order to stay clear of collisions with the COTS part.

The basic steps for the algorithm are as follows:

1. Orient the printer nozzle to be normal (or at some specified angle) to the velocity trajectory of the nozzle at all times (this yields two possible solutions for nozzle orientation).
2. Always orient the nozzle to be pointing towards the nearest COTS item (this selects between the two solutions in (1)).

In this design the path planning algorithm simplifies each COTS item to a 'keep out' zone centered at a specific point location and with a specified height. The assumption is made that the G-Code file for the printed part is designed to accurately represent the exterior profile of the COTS item, and thus no other information is needed about the COTS item's external geometry. Not relying on additional geometry information implies that the part designer avoided creating a design which violate the geometry constraints of the printer (i.e. parts are too close together, or

have concave profiles that the nozzle cannot access), since these potential collisions will not be detected by the path planning algorithm.

The path planning algorithm was implemented in MATLAB and is applied to a given G-Code file. The inputs to the algorithm are a G-Code file produced by Slic3r which encodes the XYZ locations the print nozzle must move to, and a list of COTS item locations and heights. Using this information, the algorithm computes the necessary angle of the rotation axis for each G-Code movement command, and produces a modified G-Code file. It was discovered that Skeinforge only produces straight line commands (G00 and G01) which greatly simplifies the implementation algorithm because rotation along an arc doesn't need to be considered.

The actual implementation of the algorithm is as follows:

For each G00 or G01 (straight line movement) command:

1. Compute the vector ($\vec{n}$) in the XY plane from the current position to the center point of each COTS item ($\vec{c}_i$) which is at or below the current Z position

$$\vec{n}_i = \vec{c}_i - \overrightarrow{p_1}$$

2. Determine the COTS item which is nearest to the current position.

$$\vec{n} = min_i(\|\vec{n}_i\|)$$

3. Compute the vector for the current movement command

$$\vec{m} = \vec{c}_i - \overrightarrow{p_2}, \quad where,$$
$$\overrightarrow{p_2}\ is\ the\ desired\ (x, y)\ position\ of\ the\ extruder\ at\ the\ end\ of\ the\ movement\ command$$

4. Determine which side of the print path the COTS item is on

$$side = det([\vec{m}, \vec{n}]) \quad [1]$$

$$where\ for$$
$$side > 0\text{: } COTS\ item\ is\ to\ the\ left\ of\ \vec{m}$$
$$side = 0\text{: } COTS\ item\ is\ on\ \vec{m}$$
$$side < 0\text{: } COTS\ item\ is\ to\ the\ right\ of\ \vec{m}$$

5. Compute the velocity of the nozzle in the coordinates of the print bed

$$\vec{v} = \frac{\overrightarrow{p_2} - \overrightarrow{p_1}}{\|\overrightarrow{p_2}\|\|\overrightarrow{p_1}\|}$$

6. Compute the angle between the velocity vector and the x-axis

$$\alpha = atan2(v_y, v_x), where\ v_y, v_x\ are\ the\ x, y\ components\ of\ \vec{v}$$

7. Based on which side of $\vec{m}$ the COTS item is, rotate the nozzle to be normal to the velocity vector and pointing towards the COTS item.

$$side > 0:\ r = \alpha + \frac{\pi}{2}$$
$$side = 0:\ Invalid - Collision\ will\ occur$$
$$side < 0:\ r = \alpha - \frac{\pi}{2}$$

An example of the modified G-Code is shown in Figure 18 : G-Code modified to include additional 'R' axis commands

Figures 19 and 20 show the R-Axis commands generated for a cylinder with a rod placed inside it as a COTS item. As we can see, the arrows are all pointing towards the center, thus orienting the nozzle to not collide with the COTS item. The results clearly indicate that the algorithm is successfully controlling the nozzle orientation to avoid collision with the COTS item.

```
g1 x119.200 y133.151 e1238.61271 r18.4349
g1 x118.973 y133.665 e1238.63294 r90
```

Figure 18 : G-Code modified to include additional 'R' axis commands

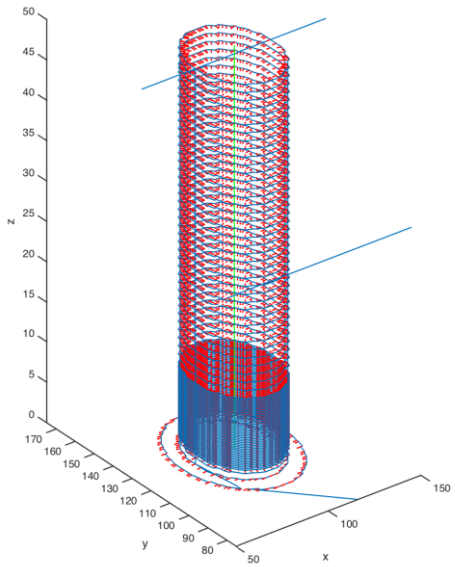


Figure 16 : Plot of modified G-Code including R-axis commands. A COTS item (represented by the green line) was specified to be at the center of the cylinder.

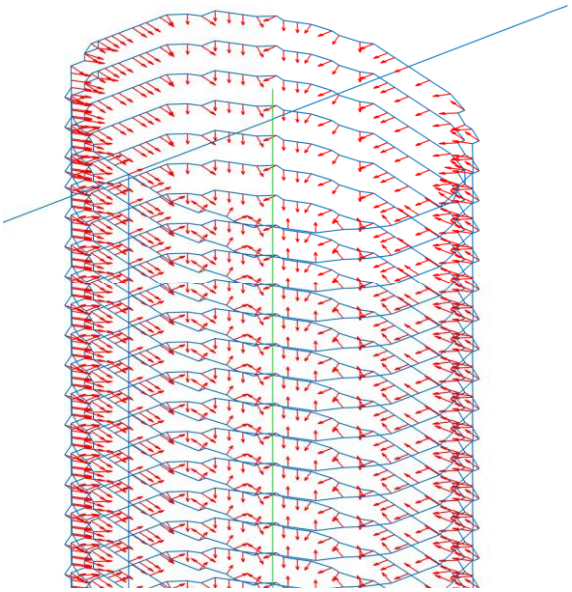


Figure 15 : Close up of G-Code path. Notice the nozzle is always oriented normal to the path and towards the COTS item.

6.4.3 Graphic User Interface

The GUI ties together all the functions written in the software and provides the user with an easy to interpret application that takes care of the special G-Code generation from start to end.
The process can be understood by the following images:

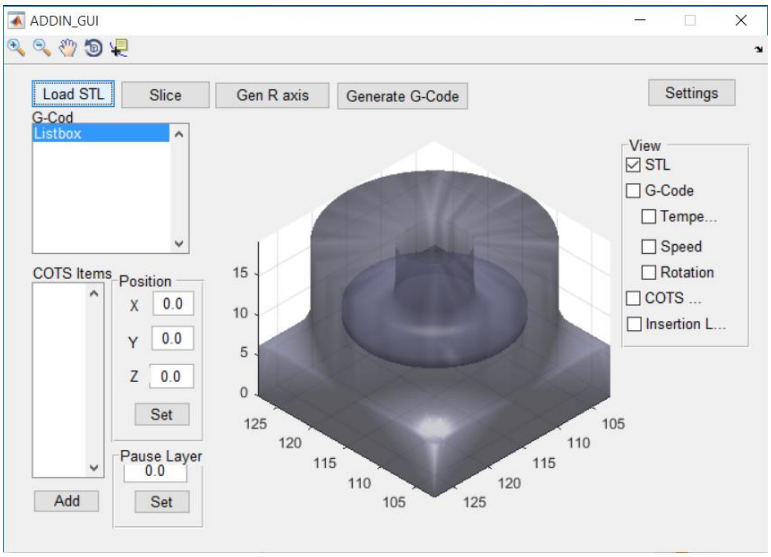


Figure 171: Visualization of STL file in GUI

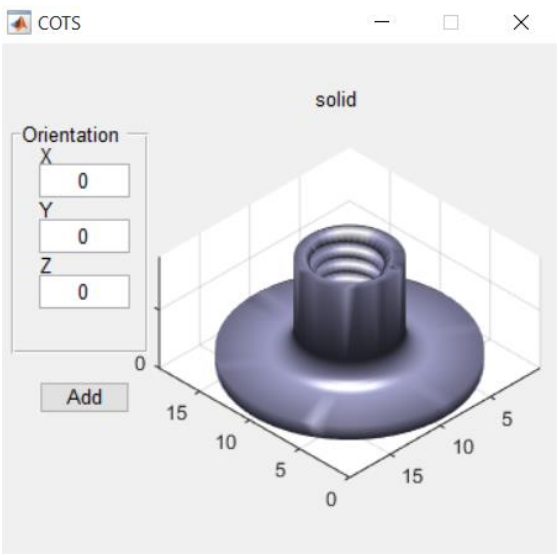


Figure 182: Addition of COTS item (with options to vary orientation)

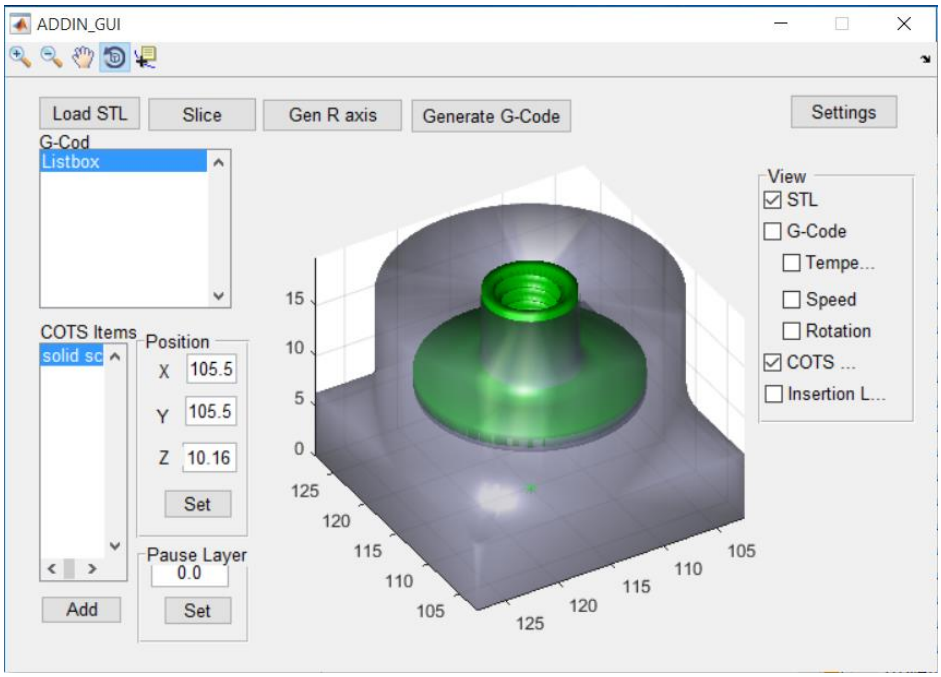


Figure 193: Positioning of COTS item within 3D part

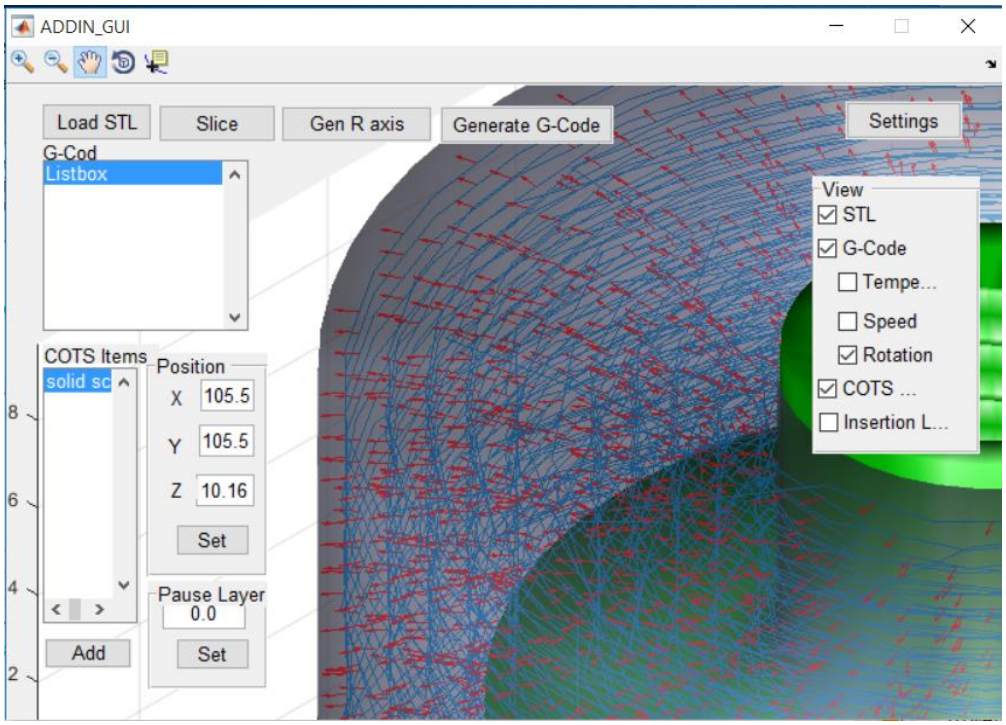


Figure 204: Visualization of R-Axis commands per G-Code

6.4.4 Settings Window

The settings window is an extremely useful functionality of the GUI that allows the user to define the commands used to invoke the slicing software locally. This makes the software completely portable and also eases the process to save settings.

As shown in the figure below, the Settings Window lets the user specify the following settings:

1. The location of their slicing software on their system locally
2. Slice Settings which vary from print to print
3. Command to invoke the slicing software from command line

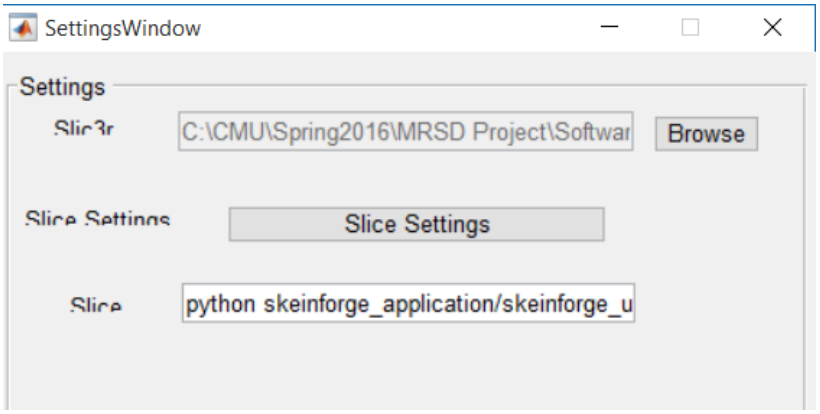


Figure 25: Settings Window

The settings window came in handy when the project required us to shift between slicing software in the last minute. When the previously employed 'Slic3r' started causing problems, we switched to 'Skeinforge' which allowed us to tune many more setting, giving us more room to obtain high quality prints. The three settings mentioned above were all we had to change in order to make this transition from Slic3r to Skeinforge.

6.4.5 Strong Points

1. The employment of classes for data handling made the software framework extremely robust. Being able to access the data in a tabular form greatly extended the capabilities of the subsystem
2. The settings window makes it very straightforward to switch between slicing software, as well as store settings pertaining to a particular print.
3. Visualization tool allows the user to place the COTS item accurately within the 3D part.

6.4.6 Weak Points

1. Currently MATLAB dependent since the web interface was not implemented.

6.5 ADD_IN

Figure 26 shows the complete integrated system ADD_IN.

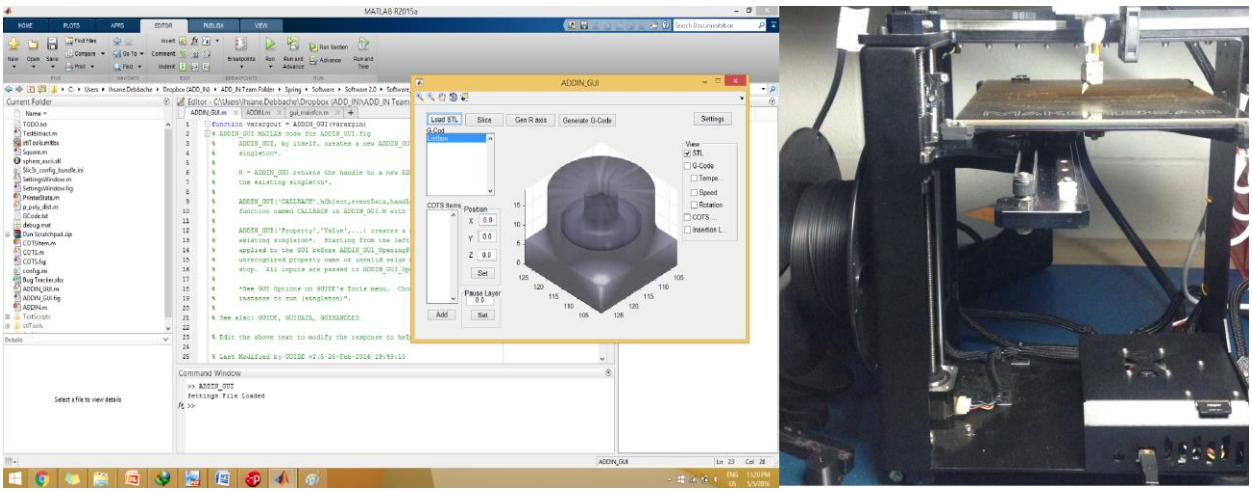


Figure 26: ADD_IN printer with software shown on the left

6.6 Spring Validation Experiments

The team's spring validation experiments will be composed of 3 parts; first we will test the functionality of the software. Second we will test the 4DOF printer. And finally we will verify that the integration of the system gives the results it was designed for, i.e., printed parts with well incorporated COTS Items that satisfy our performance requirements.

6.6.1 Objective

To verify the following:

1. The ability of the software to generate four axis G-Code commands with obstacle avoidance and insertion layer selection capability
2. The ability of the printer to correctly print by executing those commands.
3. To validate that the parts printed with ADD_IN have superior qualities than standard parts.

6.6.2 Location

NSH B506 – Team F Test Bench

6.6.3 Equipment

1. Laptop computer equipped with the following:
    a. Slicer software
    b. 3D visualization software
    c. Printer interface software
    d. *.stl* files of
        - P1: Part with stiffener rod
        - P2: Part with threaded insert
        - P3: Part with sensor/electronics
2. Reference 3D printed parts:
    a. R1: Similar to P1 but without stiffener rod
    b. R2: Similar to P2 with glued screw insert
3. ADD_IN printer configured with the following:
    a. Custom nozzle
    b. PLA filament roll
4. COTS items
5. Test bench
6. Test setup for each printed part
    a. Bending stiffness tester
    b. Pull out strength tester
    c. Electronics operation tester

6.6.4 Test Procedure:

For parts P1-P3, perform the following:

1. Use our Slicer software to load the .stl file
2. Select insertion layer and print configuration
3. Generate the *.gcode* file
4. Visualize resulting G-Code and ensure accuracy
5. Send *.gcode* file to printer
6. Start print
7. Insert COTS item when printer pauses at the insertion layer
8. Resume and finish printing
9. Conduct the following tests
    a. Bending stiffness test for P1 and R1
    b. Pullout strength test for P2 and R2
    c. Electronics/sensor functionality check for P3

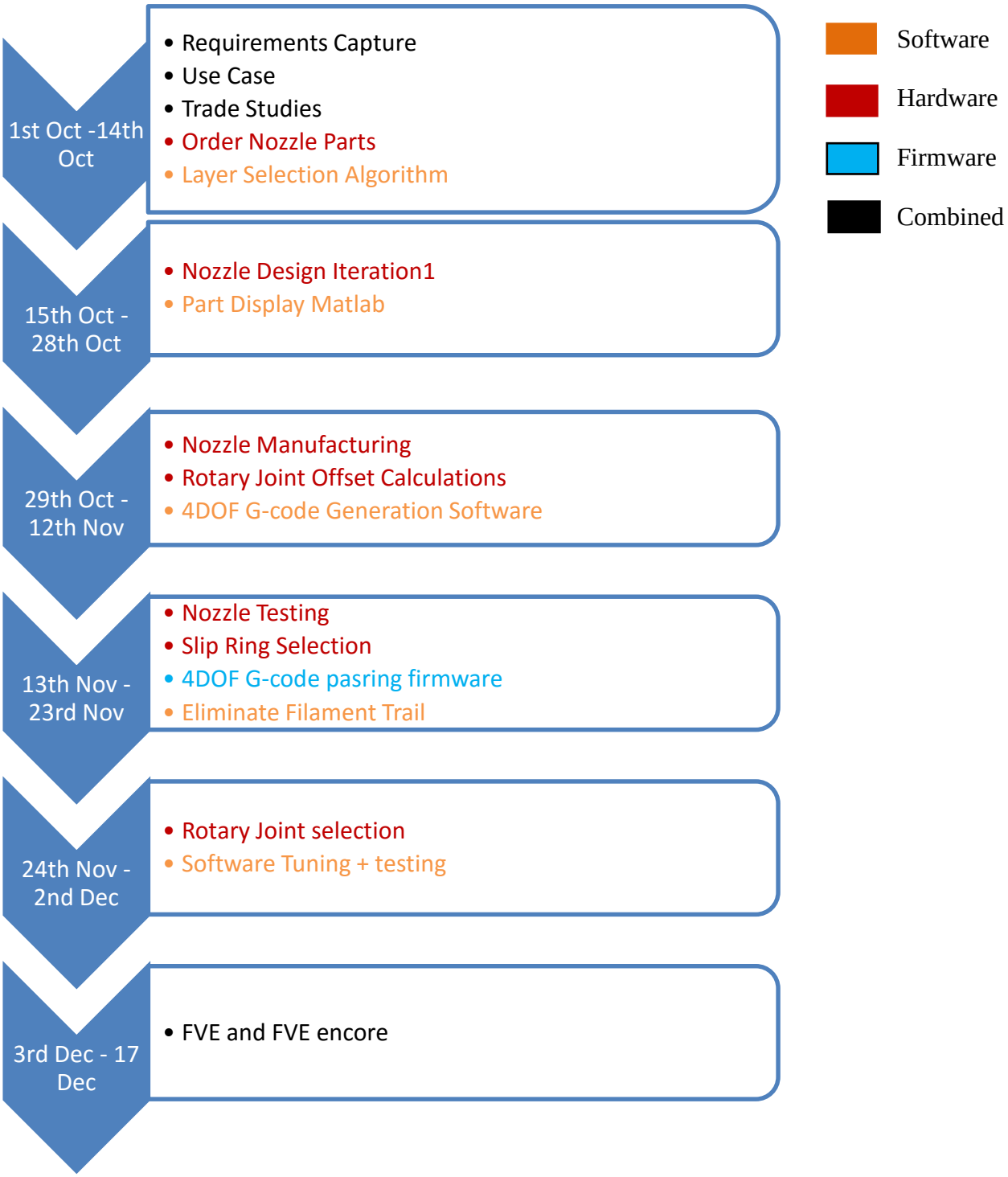
6.6.5 Success criteria

1. Generated G-code visualization should show the rotation axis being positioned to avoid collisions with the COTS item
2. Printer should successfully finish the print
3. Stiffness of P1 > Stiffness of R1
4. Pullout strength of P2 > Pullout strength of R2
5. Electronics/sensor in P3 should be functional

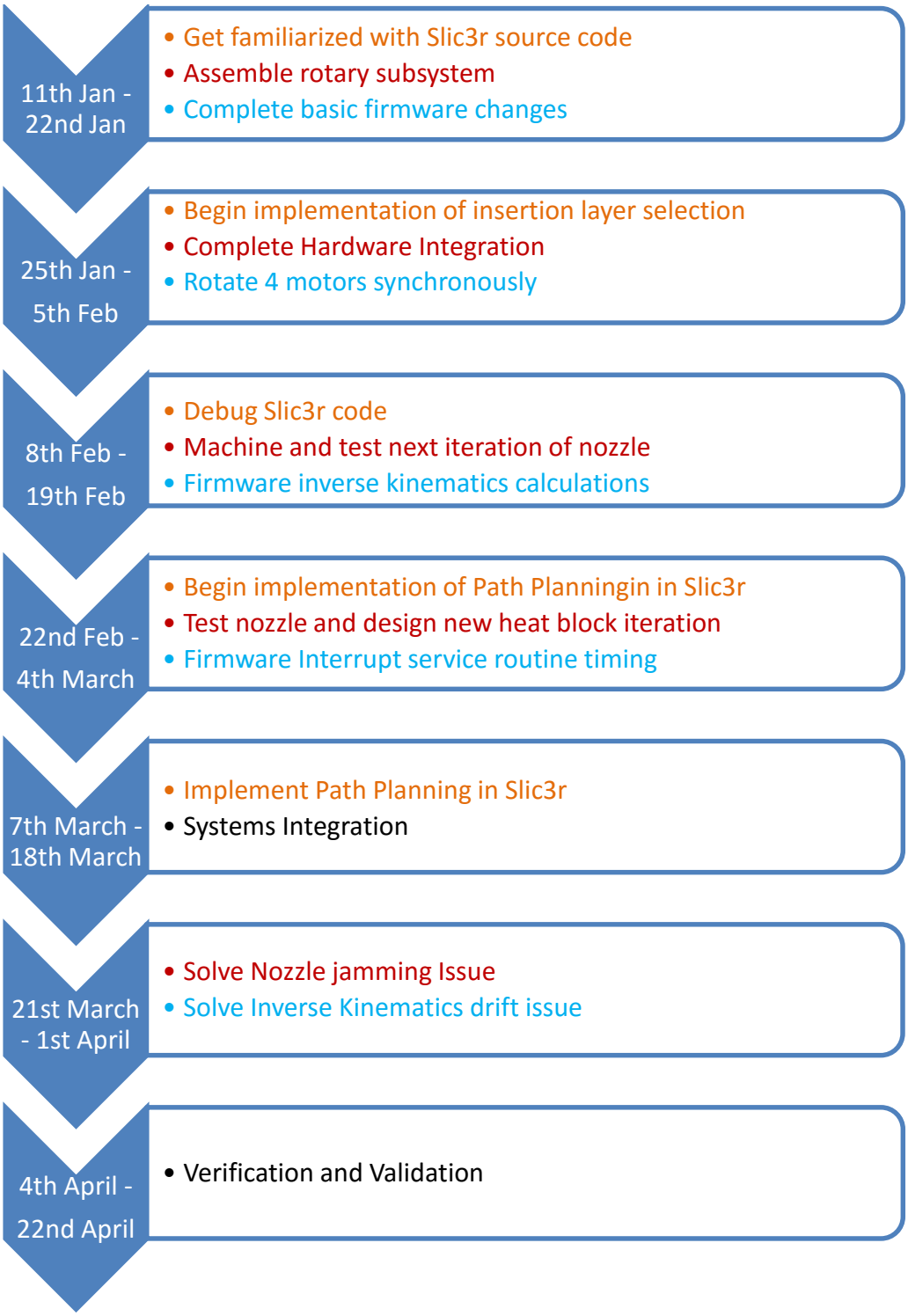
7 Project Management

7.1 Project Schedule

7.1.1 Biweekly tasks for fall



7.1.2 Biweekly tasks for spring



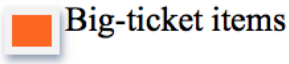
The schedule for the fall semester had relatively lower efforts when compared to the spring semester. The fall semester had definite task which could be completed in the set duration. In the spring semester the tasks had a lot of dependencies which resulted in unexpected delays increasing the workload on the project. However, the project was completed on time and a few stretch goals were also accomplished.

7.2 Budget

The final budget is described in Table 5, with a more detailed part list from the ordering excel sheet available in the appendices.

Table 5: Project Budget

Item	Quantity	Cost per unit	Total Price	Details
Second 3D printer	1	$1825	$1825	Second 3D printer to speed up the testing process
Hollow shaft stepper motor	2	$170	$340	Custom made motor from Linengineering
Slip ring	2	$150	$300	504-0800 Hollow shaft slip ring from Orbex.
Rambo backup board	2	$180	$360	Rambo v1.3 board
Makerfaire tickets	4	$20	$80	Tickets for the Makerfaire to do market research
Backup extruder motor	1	$80	$80	Kysan geared stepper motor with drive gear
V3b hot end kit	1	$65	$65	V3b hot end for testing and backup
McMaster parts	N/A	$240	$240	Various machining parts for the extruder and COTS parts for testing
Hot end parts	N/A	$250	$260	Many Barrels, nozzles, heat blocks, groove mounts , thermistors and heaters.
Flexible filament	1	$52	$52	Ninja Flex filament for testing
Extruder wiring harness	1	$30	$30	Wiring harness for the added motor
Backup Print bed	1	$24	$24	Backup glass print surface
Sensors	6	$5	$30	Optical Sensors for End-switch
Total budget			$3676	



7.3 Risk Management

Recognizing and mitigating risks is essential to avoid unexpected failures, changes, and delays to the project. The ADD_IN team has adopted a systematic method for tracking and managing risks.

7.3.1 Risk Management Process

Risk Identification

Risks or all magnitudes can be identified by all team members at all times. Tracking additional risks incurs almost no burden and in many cases risks which were initially considered to be minor can grow to become significant. Risk brainstorming sessions are occasionally held, but most risks are identified on a day-to-day basis while working on developing the system.

Risk Level Assessment

To consistently asses the magnitude of a risk, we developed rubrics that define a scale for the likelihood and severity of each risk. These rubrics are depicted in Table 6 and Table 7.

Table 6 : Risk Likelihood Legend

Legend – Likelihood	
1	Highly Unlikely - Almost impossible
2	Unlikely but possible to occur
3	Likely to occur sometime during the life of the project
4	Expected to occur during the life of the project
5	Estimated to occur

Table 7 : Risk Severity Legend

Legend-Severity	
1	Inconsequential hardware loss, Time delay < 1 day
2	Minor Hardware loss. Time delay ~1 day
3	Hardware loss. Time delay 1-5 days. Inability to meet non-critical requirement
4	Inability to meet critical requirement (significant reduction in printer's capability)
5	Failure of project/All major requirements

After assigning an appropriate likelihood and severity level to each risk, the risk is added to a risk tracking excel template. The template calculates and ranks all risks according to their total risk level (product of likelihood and severity). The results are broadly classified into low, medium, and high risks based on Table 7 : Risk Severity LegendMitigation strategies are then immediately implemented for all high and medium level risks.

Table 8: Total Risk Level Definitions

	Legend – Total
0-6	Low Risk
6-12	Medium Risk
12+	High Risk

7.3.2 Top Risks at CDR

At the time of the critical design review the ADDIN team was tracking twenty risks, ranging in total risk level from two through six. The top four risks are shown in Table 9 and indicated on a risk matrix in Figure27.

Table 9: Top 4 risks being tracked

ID	Type	Description	Likelihood	Severity	Total	Owner
2	HW	Rotary joint not sufficiently precise	2	3	6	Ihsane
13	HW	Unforeseen challenges in rotary joint mounting	3	2	6	Nikhil
20	HW	Custom stepper motor wrong specifications (Est lead time ~6 weeks)	2	3	6	Dan
5	SW	Slic3r cannot be modified	2	3	6	Astha

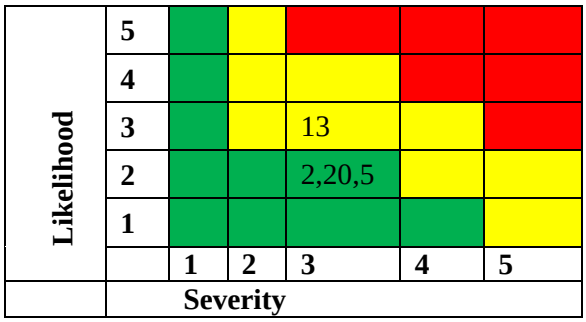


Figure 27: Risk Matrix at CDR

7.3.3 Top Risks prior to SVE

At the time of the SVE the ADDIN team was tracking twenty six risks, ranging in total risk level from two through six. The top four risks are shown in Table 0 and indicated on a risk matrix in Figure28.

Table 40: Top 4 risks being tracked

ID	Type	Description	Likelihood	Severity	Total	Owner
21	HW	Failure of slip rings	3	4	12	Ihsane
24	HW	Nozzle jamming	3	2	6	Dan
22	FW	Firmware drift	2	3	6	Nikhil
26	SW	Software caused COTS collisions	2	2	4	Astha

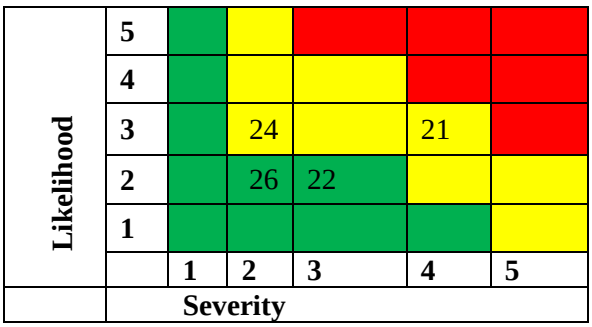


Figure 28: Risk Matrix at SVE

8 Conclusions

8.1 Key Lessons Learned

There are several lessons we learned this semester, some related to our project and some about team work. The key lessons learned include:

- Leave ample time for testing
- Printing is a slow process and requires many runs in order to arrive at the correct parameter settings.
- The printer is fragile so treat it with care
- Nozzle's Jam!! Use Canola oil to avoid jamming
- Stock up on spare parts
- Ensure that all team members have the same version of software
- Software compatibility caused some problems this semester. At one point of time, all the team members were using different versions of Slic3r. This caused quite a bit of confusion until we all got back on the same page.
- Firmware is hard to debug, so use debuggers from the start of the project where possible
- A team that eats together sticks together
- It was good on the team to have a spare printer and so use all your budget
- OOPS concepts are good to incorporate

8.2 Future Work

The following work will be undertaken to get this project to a better platform:

- A new nozzle has been designed which will be 3D printed
- Implement a web platform for the project
- Modify the software to plan around multiple COTs
- Modify software to control extrusion rates for complex infill paterns

9 References

[1] Slic3r software: http://slic3r.org/
[2] Makergear M2 3D printer: http://www.makergear.com/products/m-series-3d-printers
[3] Cura software: https://ultimaker.com/en/products/cura-software
[4] Tang, Tran Duc, "Algorithms for collision detection and avoidance for five-axis NC machining: A state of the art review" J. Computer Aided-Design pp.1-16, 2014.
[5] Rambo Development Board: http://reprap.org/wiki/Rambo_v1.1