

CprE / EE / SE 492
Team MAY14-07
3D Printer Software Programming
Final Document

Team Members:

Arielle Czalbowski	ariellec@iastate.edu
Piriya Kristofer Hall	pkhall@iastate.edu
Albert Kurniawan	albertk@iastate.edu
Wanting Zhao	wzhao@iastate.edu

Adviser / Client:

Dr. Thomas Daniels	daniels@iastate.edu
--------------------	---------------------

PROJECT DESIGN

I. System Requirements

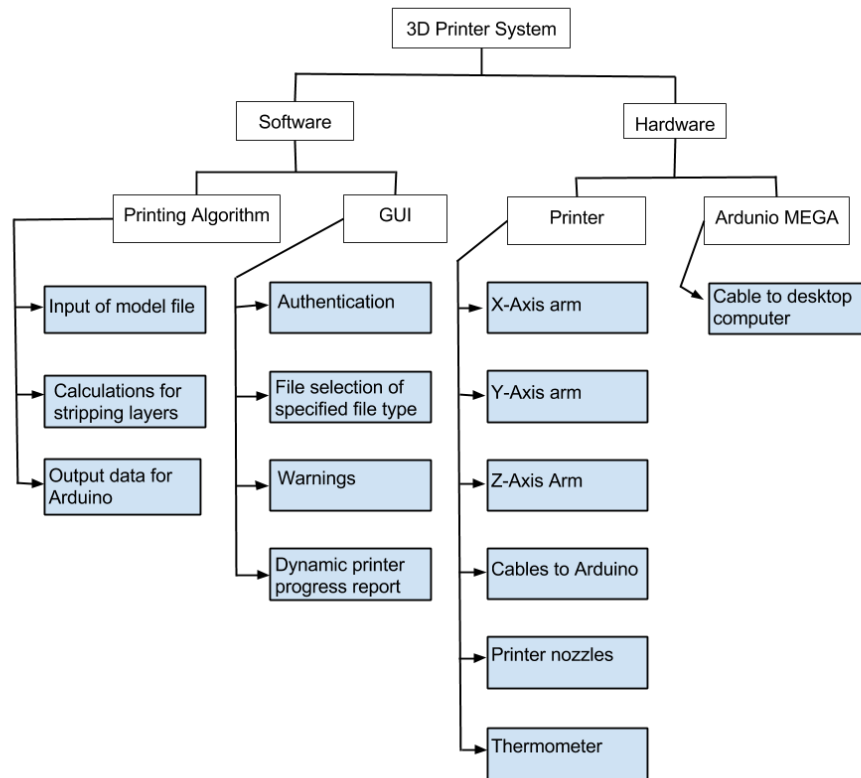
Functional Requirements

- The product shall take as input a model file that is compatible with our program, and will otherwise produce an error.
- The product shall determine a path that the printer nozzle can take in order to print the 3D object.
- The product shall produce a path for the printer head to follow.
- The product shall send the information of the printer head path to the Arduino MEGA via USB.
- The product shall only accept one model file at a time.
- The product shall not allow the user to manually move the printer via commands excepting maintenance and calibration.
- The product shall be able to pause printing if there is inadequate filament and allow the user to replace the filament cartridge.
- The product shall allow the user to cancel the print job at any time.
- The product shall take users' payment information and write it to a text file.
- The product shall take users' print information for a given print and write it to a text file.
- The product shall print the plastic model file onto a heat plate.
- The product shall only print when the heat plate is hot enough for the print job to be successful and without significant error.
- The product shall regulate the temperature of the heat plate.
- The product shall give power to the heat plate if it is deemed to be under a printable temperature.
- The product shall reduce power to the heat plate if it is deemed to be over a printable temperature.

Non-Functional Requirements

- The product shall take the average user no longer than 10 seconds to figure out the interface.
- The product shall produce an error message due to software programming no more than 10% of usage time over a 3 month period.
- The product shall be easy to use by anyone with at least a high school understanding of English.
- The product shall be easy to use by someone who has had no more than 6 months of exposure to computers.
- The product shall take no more than 1 minute to complete the path finding algorithm per layer and send it to the Arduino MEGA.
- The product shall produce a faulty print during no more than 5% of usage time over a 3 month period.
- The product shall not allow a model file over 1 GB.
- The product shall always clearly warn users of the safety concerns of utilizing the 3D printer.
- The product shall provide clear instructions for preparing the printer hardware.
- The product shall only allow Iowa State University students and faculty access to printing, and will therefore always require authentication.
- The product shall display the current temperature of the heat plate on the user interface screen 90% of the time.
- The product shall take no more than 10 seconds to read input from the thermometer.
- The product shall always clearly warn users of the safety concerns of working with the heat plate.
- The product shall inform the user of how long the print has lasted.
- The product shall require the user to enter contact information before being able to utilize the 3D printer to provide payment information for the print.
- The product shall not allow the user to manually adjust the thermometer.

II. Functional Decomposition Diagram



III. System Analysis

i. Scope Definition

The scope of this project is to produce software that will allow an ordinary desktop computer to interface with a 3D printer and produce a 3D model. This will be done through software we write, which will utilize G Code conversion so that an Arduino board can communicate with a 3D printer. Within the scope of our work is the actual software that does the communication, as well as a GUI and a calibration program. Also within the scope of our project is control of a heat plate. This interface involves gathering, displaying data from, and dynamically parsing input for the program. Outside the scope of our program are the specifics and technicalities of the hardware that we are interfacing with, including the setup for the

hardware.

ii. Problem Analysis

The overall problem that we are facing is our ability to create a piece of software that will allow the user to easily produce a 3D object via a 3D printer from a .STL file.

One portion of this software requires us to deconstruct the .STL file in such a way that the hardware will be able to understand the commands that it is being fed and will respond to them appropriately in order to accurately print a 3D object. To this end, we will be utilizing the model slicing software of the RepRap CAM software stack, called Slic3r, to create a method for the software to communicate with the hardware that will be printing the object.

Another portion of our software will allow the user to interact with the graphical user interface in order to control the printer's software. The user must be able to select an appropriate model file and control the printer's movement, both for convenience and for safety.

We also need to have a thermometer that can interface with the printer. The program must dynamically send back the current temperature of the heat plate to the user so that, if there are any major problems, the user will know if action needs to be taken.

Finally, the additional program that we are writing, a program to calibrate the hardware for the 3D printer, must be able to provide an accurate and useful method of calibration data so that any print jobs will be successful.

iii. Decision Analysis

We have come to the following decisions about the direction our project should take, as well as the rationale of why they should be taken:

- We have chosen to write the calibration program in a method such that it will run a pre-set certain number of times. This is because allowing the program to run a large set number of times will make it simpler for the hardware team that we are working with to calibrate the printer.
- We have chosen to implement a graphical user interface that works solely with buttons in order to simplify the program and make it easier for people without strong backgrounds in computers to use.
- We have chosen to display the live output of the 3D printer in order to make it easier for users to see the status of the 3D printer and be aware of any errors in an easy-to-read manner.
- We have chosen to utilize the RepRap software stack because it has a lot of pre-existing tools for 3D printing that will aid us in creating adequate software for the printer.

iv. Standards

The project that we are working on will not see commercial use, and instead will be used by students and faculty. In addition, our project will be a continual work in progress due to changing and improving technology as 3D printing becomes more popularized. Because of this, our standards are limited. However, there are still some standards that we should follow that will need to be taken into account.

- ANSI ISO 10005 / 10006 / 10007 Quality Management Standards. *Due to the fact that we are working on a 3D printer, which is moderately new technology, the item that we are programming does not have well-set standards. By examining past standards for 3D printers and ensuring we meet them, as well as potentially setting some standards of our own, we will ensure that our 3D printer fits quality as well as potentially setting the standards for other heavily machined 3D printers that will come after ours.*

- Underwriters Laboratories Standards. *This is due to the fact that our printer has a heavy hardware component and can be very dangerous.*
 - Standard 2785 (Printing cartridges)
 - Standard 796 (Printed-wiring boards)
 - Standard 746F (Polymeric materials)

IMPLEMENTATION DETAILS

I. **Input / Output Specification**

Our system's input will be a model file in .STL format so that it can be sliced properly to be sent to the 3D printer. The software that we write will not take any other input; the printer will be restricted to an STL-format file in order to function properly.

Our system's output will be data that can be sent to the Arduino in order to control the 3D printer. The data will direct the 3D printer to print a plastic model of the file that was sent in as an STL file.

Our system has an additional, intangible output: Our program will show dynamic data on the user interface of the status of the printer as well as on the status of the thermometer. The repeated output will be shown prominently on the user interface to inform the user of the current printer's work and any adjustment that the program is making to the printing.

II. **User Interface Specification**

The user interface will be effectively utilized in order to allow the user an easy and effective way to communicate with the 3D printer and accomplish the task of printing a 3D object without extensive or prior experience in computer technology.

The user interface will consist of a number of buttons as well as a text area for feedback. The components of the user interface include:

- A warning message that will alert the user of the dangers of the 3D printer and require them to agree to the dangers of the printer before they can use the program.
- A form that the user must fill out with contact information so that they can be contacted to provide payment after using the 3D printer's capabilities.
- A button that will allow the user to select a .STL format file from their hard drive in order to print. The user may select a file as many times as they wish, but once printing has begun, the user may not select another file.
- A button that will allow the user to print a file. Once pressed, this will begin the slicing of the model, which produces the G Code representation of the model, then the sending of the G Code over USB to the Arduino MEGA.
- A button that will allow the user to cancel their current print job. This will stop the printer from continuing its print and terminate the job immediately.
- A button that will allow the user to pause their current print. This will suspend the printer's path indefinitely until the user chooses to resume the print job, in which case the 3D printer will continue its print job exactly where it left off.
- A button that will allow the user to restart the current print job. This will stop the user's print job and go back to the beginning of the code, where it will start the print over again.
- A window that will display dynamic feedback from the printer. As the user prints their file, they will see a live feed of the status of the printer and what the printer has accomplished. The temperature of the heat plate will also be displayed. Any errors that occur during this print, as well as confirmation messages, will also appear in this box.

III. Hardware Specification

Due to the fact that we are largely doing a software-based product, the hardware that we provide is necessary for the peripheral functionality of the software rather than the core functionality:

- A desktop or laptop containing:
 1. At least one (1) USB port
 2. A modern operating system
 - a. This includes: Windows XP or above, Mac OS X, or an updated Linux distro (Ubuntu, RedHat, etc.)
- A USB cable that will interface between the desktop or laptop and the Arduino board. This USB cable must have a male Standard-A USB port on one end, and a male standard-B USB port on the other end.
- A thermometer that will monitor the heat of the heat plate.

IV. Software Specification

The software specifications for our project are as follows:

- A calibration program that must:
 1. Run a number of specified times in a set pattern.
 2. Run easily as to allow visual calibration of the hardware.
- A 3D printing program that must:
 1. Prepare the 3D printer to receive a print job.
 2. Take .STL file as input.
 3. Allow the user full control over what the printer is doing.
 4. Not allow the user to manually move the printer.
 5. Not allow the user to interfere with the print job.
 6. Send G Code over to the Arduino so that it can control the 3D printer.
 7. Output information to the user via the program interface on print status.
 8. Output a 3D-printed version of a model file.
 9. Interface with a thermometer and return the value of the temperature to the user.

V. Languages Used

The following languages are used in our implementation:

- C# is utilized to program the GUI.
- C++ is utilized to program the firmware.

VI. Firmware Used

- Repetier is used to interface with the printer.

TESTING PROCESS & RESULTS

I. What we are testing

We are planning to test the three pieces of software that we are planning to deliver:

- a. A program to calibrate the 3D printer's hardware.
- b. A program that will accept an STL file as input and output a 3D-printed item by slicing the model into layers, saving the layers in G Code, then sending the G Code to the Arduino via USB.
 - i. This program must successfully interface with a thermometer and must accurately read the temperature of the hot plate and feed the data back to the user in a timely manner.
- c. A program to act as a GUI that the user can interact with. This will allow the user to select a .STL file, print, pause print, and cancel print. The user must also enter their information to be contacted later for payment. Before being allowed to do any of this, the user must agree to a warning of the dangers of using the 3D printer.

II. What we are not testing

While the following items are related to our 3D printer, they are outside the scope of our software and therefore out of the scope of our test area:

- i. The physical circuit boards for the 3D printer.
- ii. The hardware that the 3D printer itself is composed of.
- iii. The physical Arduino board that the 3D printer is connected to.

- iv. The physical stability of the hot plate and its hardware setup.

III. Major areas to test

While fully testing all aspects of our software is important, we feel that the following are the most important items to test:

- i. Whether the printer receives all of the commands correctly and in a timely manner.
- ii. Whether the printer produces a viable 3D model.
- iii. Whether the software we write communicates effectively with the printer hardware.
- iv. Whether the software we write communicates effectively and accurately with the thermometer.
- v. Whether the user interface successfully communicates to print.

IV. Prototyping / Testing Process

We plan on having three different types of software to prototype and test.

a. Calibration Program

The first kind of prototype we will deliver is a calibration program for the hardware team, which will allow them to determine the accuracy of the 3D printer. Specifically, with this program, the accuracy of the stepper motor will be testable, which will allow us to determine the types and accuracy of models that we may do.

We plan to test the calibration program by having it run a small number of times and determine whether or not the printer moves in the way that we expect it to.

b. Back-end Program

The second prototype that we will deliver is a prototype of the program that will send G Code to the Arduino where it will interpret the G Code to move the 3D printer.

We initially plan to test this program by attaching something inexpensive, such as a marker, to the print head so that more expensive plastic is not wasted. After the initial test, we plan to run the same tests, but by utilizing the actual plastic.

The tests that we perform on this program include testing how well the 3D program can handle printing one layer through a simple STL model file. Once the printer has demonstrated its ability to handle single layers, the entire STL file will be used in order to determine how well the 3D printer can produce a full model.

c. *GUI*

The third prototype that we will deliver is a prototype of the GUI. This is the GUI that will provide the warnings to the user, require the user to enter payment information, and finally, allow the user to select their STL files to print.

We will test this program by running through each of the buttons and functionalities and determine whether they work the way that they are expected. We will also show the GUI to the client to determine whether it matches with our client's vision, and make changes accordingly.

d. *Overall Testing*

When all of these pieces are together, we are going to do overall testing. After we have determined that the program will theoretically print a 3D model correctly, we will attach the heat plate and begin working on interfacing the printer with the heat plate, and test whether the heat plate is able to successfully interface with the hardware and heat up and cool down as it should.

We will also test whether the GUI can accurately interface with the

hardware. We will test this by linking the back end printing functionality to the buttons on the GUI and test the buttons to see whether or not they produce the result that is expected.

Once these tests have been deemed successful, we will test whether our code dynamically works with everything. We will also test how real 3D models work with the printer and whether these 3D models have any problems printing.

V. Testing Results

a. Calibration Program

Our calibration program was successful. We were able to produce a calibration program that allows the user to input how many repetitions they want the arms to run through.

The calibration program ran the printer in the way that we've expected, and it has proved useful for calibration purposes.

b. Back-end Program

The COM port was able to be opened and communication could be established between the host computer and the Arduino. The program performs the print a bit slower than the host software provided by Repetier, but that is due to differences in design.

The Repetier firmware works as expected, as it performs the movements and heating of the heated elements that were sent to it via USB.

c. GUI

Our GUI program has had some moderate success. Not all of the functionality is easy to achieve in C#, such as making timing clocks and writing to text files. There are still problems communicating with the COM port. These functionalities will be continued in future iterations of the project.

However, other parts of our GUI have met with great success. We have been able to successfully produce the warnings, gather information from users, and produce an interface for printing. The user can only select files that we allow with strict error-checking to ensure that the user cannot input an invalid file, as well. The buttons are all functioning, and the dynamic information feed works correctly.

The program also has smooth transitions between forms, and ensures that it smartly utilizes memory and has proper termination to allow the user's computer to run smoothly.

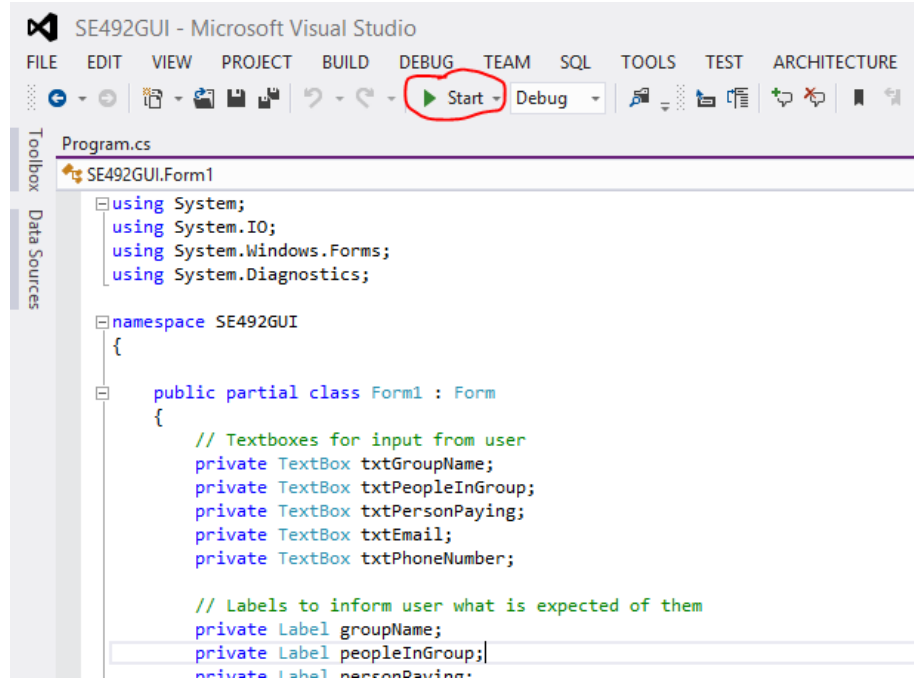
d. Overall Testing

Test printing of a simple and complex model produced the desired result. The heat bed, and other heated elements were heated correctly and to the right temperature. The simple model was a simple cube, and the complex model is a model of the Washington monument. The GUI was able to interface with the printer correctly, and the buttons do as they should.

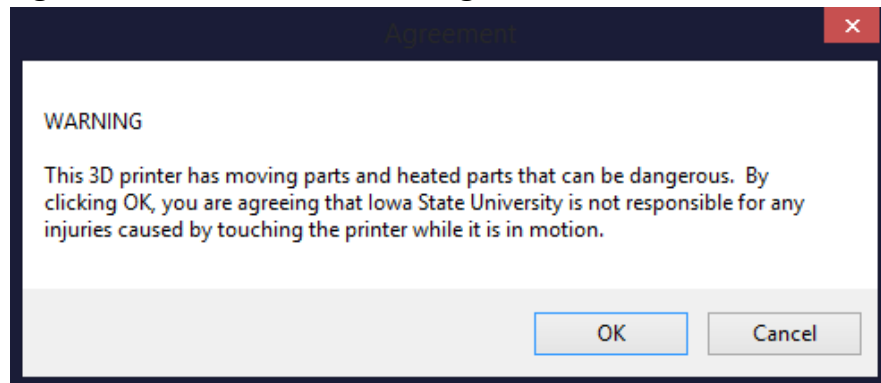
APPENDIX I: OPERATION MANUAL

I. System Setup

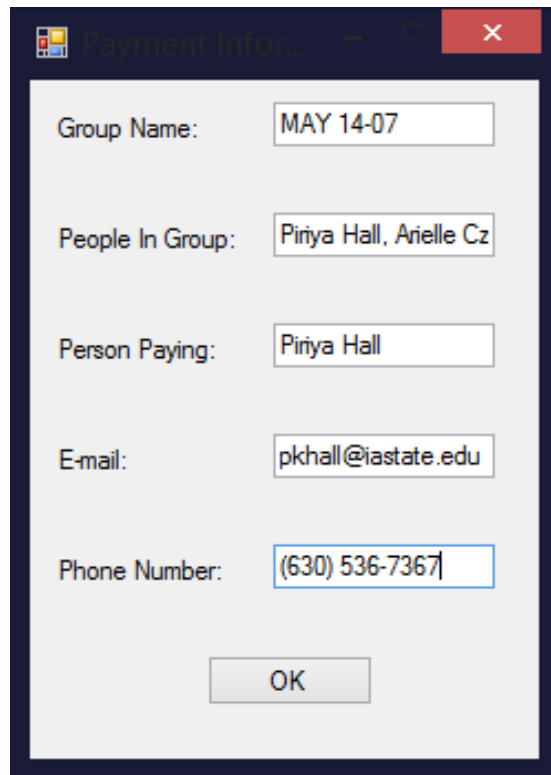
1. Run program to begin GUI.



2. Agree to or refuse the warning.



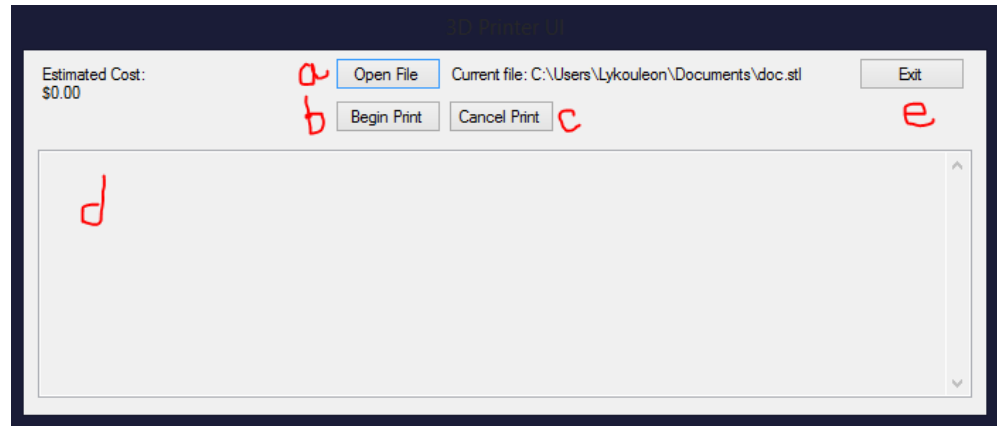
3. Fill information into form and hit “OK”.



The image shows a screenshot of a Windows-style dialog box titled "Payment Info". The dialog box has a dark blue title bar with a close button (X) in the top right corner. The main area is light gray and contains five text input fields, each with a label to its left. The fields are: "Group Name" with the value "MAY 14-07", "People In Group" with the value "Piriya Hall, Arielle Cz", "Person Paying" with the value "Piriya Hall", "E-mail" with the value "pkhall@iastate.edu", and "Phone Number" with the value "(630) 536-7367". At the bottom center of the dialog box is a single "OK" button.

4. Open file. Components of GUI are as follows:

- a. Open file: Open the .STL file that is to be printed.
- b. Begin print: Begin printing selected file. This button will change to “pause print” on beginning a print.
- c. Cancel print: Cancel current print job. This will end the current print and allow the user to select a new file.
- d. Print log: Displays the status of the printer and the log of the actions the printer has taken.
- e. Exit: Exits the program.



II. System Demo / Test

In order to test the system, follow the instructions above to run the machine. Choose an STL file to input to the machine, and click print. The back end of the machine will run, and the item that is chosen will be produced.

The 3D printer takes a long time to print the object, so patience is necessary in order to see a print to completion.

APPENDIX II: ALTERNATIVE DESIGNS

I. **Alternative Firmware**

a. SPrinter

- i. One of the first firmware designs that we considered was SPrinter. However, SPrinter did not work with our hardware, so we chose not to use it in our final design due to its incompatibility.

b. Marlin

- i. Another firmware design that we explored for our project was Marlin. Marlin had a lot of aspects with it that worked well with our printer project; however, it would not heat the heat bed properly, which meant that our plastic molds would have trouble forming properly. Marlin was also not moving the axes correctly, which led to the breaking of some of the couplers in the printer. Therefore, we had to forego Marlin in favour of a firmware that would properly work with our design.

c. Repetier

- i. Although Repetier was the firmware that we ultimately ended up using, we went through numerous iterations of Repetier before we were able to decide on the current one that we utilized. There were a number of problems with the various versions of Repetier that we explored.

Some version of Repetier that we tried to use would not properly move the printer arms, which disallowed us from properly printing anything at all. Other versions of Repetier that we tried did not heat the bed correctly, which would prevent our plastic molds from forming properly.

The versions of Repetier that we used were not specific versions; rather, we explored various custom versions that we

felt would help us best. These were the iterations that we went through that did not work for the purpose of our printer, and each had their own set of challenges that caused us to ultimately abandon them.

II. Alternative Coding

An alternate GUI programming language that we considered early on was the QT library for C++. Initially, we felt that this would be a good library to use because we had initially planned to program the majority of the project in C++, and had a need for a language that would allow us to create visual GUI components.

However, as we continued to work on the project, we found QT to be very difficult to work with. The tutorials that were online seemed sparse and uninformative for the purposes that we needed to build the GUI, and the language felt a bit counterintuitive. Additionally, QT required a special programming environment. The tutorials that were given were difficult to make work in the particular environment, and the Visual Studio plugin for QT had a number of difficulties and we were ultimately unable to get it to work.

Fairly early on in the design process, we chose to switch to C# to program our GUI in. C# was incredibly intuitive and worked well with the Windows operating environment that the 3D printer must use. Its built-in library is extensive and suits our needs well, and no additional libraries or considerations need to be taken into account in order to get it to work. Since we were able to find a better language to program our GUI in, C++ and QT were scrapped.

III. Alternative Design

a. GUI

i. Login Screen

Originally, we were planning to have a login screen for our GUI to ensure that only students and faculty of Iowa State University would be able to use the 3D printer. However, this

idea was abandoned since access to Iowa State's user and password database did not seem feasible. In addition, we felt that the functionality of the login screen was already present, since the user must log onto the lab computer with their Iowa State University username and password to gain access to the program, which would therefore already act as an authentication to use the 3D printer.

ii. GUI Buttons

The GUI originally was going to have three separate buttons for beginning a print, pausing a print, and cancelling a print. However, we felt that this could be confusing, and it would be easy for a user to click the wrong button, which could cause errors in their print and possibly waste money. Therefore, we decided to change the buttons to be more intuitive and easier to understand in a way that better mimics existing print programs. The three buttons are still functionally in use, but the manner in which the buttons are used and how they appear to the user underwent a drastic change.

b. Backend

i. Firmware

Originally, the Arduino that was used was an Arduino DUE. Firmware would all have to be completely made from scratch, along with all of the hardware circuitry to handle the heating of all the heated elements.

ii. Communication

The backend program that communicates with the Arduino was originally envisioned to convert G Code into a custom instruction set for the Arduino DUE and the custom firmware to interpret. This was scrapped when the decision was made to switch from the DUE to the MEGA, and the use of the RAMPS board to handle the controlling of the heated elements and movement of the axes.

APPENDIX III: OTHER CONSIDERATIONS

I. Lessons Learned

- a. Working between two team poses a number of challenges that we had not initially anticipated during the course of our senior design project. Our Senior Design project was shared between two teams: Team May14-06 and Team May14-07. The project was divided in that our team, May14-07 was tasked with designing and implementing the software for the 3D printer's functionality, and our partner team, May14-06 was tasked with designing and implementing the hardware for our team.

Working with another team posed many unique challenges, and our two teams had to learn a lot about the dynamics of our team, our unique responsibilities, and other like information in order to make the marriage of our projects successful. The four most important challenges we faced and lessons we learned regarding our team dynamic were sharing workspace, relaying the project back and forth, miscommunication problems between our teams, and how to resolve those problems with constant communication.

i. Sharing workspace

Our teams both required use of the 3D printer to test our work. Team May14-06 required direct access to the 3D printer in order to test the hardware that they implemented and circuits that they were trying to build among other hardware-related projects, as well as troubleshooting issues that arose. On the other hand, our team also required the workspace to work on writing code for the 3D printer as well as test the code that we had programmed.

The need for both of our teams to simultaneously use the physical 3D printer was a challenge that was difficult to resolve, since both of our teams wanted the ability to work on the

printer at the same time and test our work. Since neither of our teams worked on a particular schedule and time spent working depended on the urgency of the work as well as the stage in which each team was at on their current project, sometimes each team would find that the other team was in the lab when they wanted to work.

Although a clear-cut solution for this problem was difficult to come by, a method that we used to ease the burden was to make team-specific meeting times that each group would attend that would guarantee time to work. Although these team meetings tended to last around an hour, it guaranteed our teams time to group and work with the printer. In the future, we would likely take care of this problem by speaking more with the other team and setting up specific work times so that each team could be guaranteed the printer for an adequate number of hours per week to get a significant portion of work done uninterrupted and reduce fruitless trips to the lab to work with the printer.

ii. Relaying the project back and forth

An issue that we encountered during the course of this project was relaying the project back and forth; that is, some portions of Team May14-06's project depended on us finishing something in our own project, and conversely, some portions of our project depended on Team May14-06 finishing parts of their own project.

This point caused a lot of problems because it happened that, especially in the beginning, we could not work on our portion of our project because we needed to wait for the other team to finish something. Of course, the other team had the same issue. The biggest consequence that arose was that we were unable to make progress on our portion of the project. This would cause delays in both design and testing. We were able

to do some testing and implementation on our own computers; however, this was not necessarily the case. The ability to use the computer directly connected to the printer was necessary for the software that it contained and, most importantly, its direct interface to the printer. Access to the printer itself and all requisite software is mandatory for a majority of software testing. We were sometimes unable to test the software due to our inability to access the printer, which hindered our own progress.

The waits that we faced were not necessarily due to Team May 14-06's status on their current portion of their project. On occasion, hardware would break. This would cause both our team and the other team pause in our project design since Team May 14-06 would have to stop their work to fix the printer, and we would be able to get any work done until the printer was fixed.

The solution we had for this problem largely consisted of constant communication and relaying information back and forth to each other. We needed to communicate the status of each others' projects in addition to explaining how the other team's part of the project impacted ours. This allowed us to come to an understanding on each others' statuses and which portions of the project were most urgent to complete. It also allowed us to communicate when certain components were in need of repair, which assisted us in not exacerbating a broken printer's current condition. If these problems were to happen again in the future, I do not foresee a drastically different method of handling the situation. However, if a similar situation were to arise in the future, it would be helpful to map out all of the components of our project and go over them with our partner team so that more planning could be done in regards to which parts of the project are most reliant on the other team's and which portions of the project can be done

more individually to maximize work efficiency.

iii. Miscommunication and misunderstanding

Throughout the course of our project, there were a number of miscommunications and misunderstandings that hindered our progress, as well as a gap in understanding the semantics of the project.

Our project had its own share of miscommunications, which included information about the hardware, what the printer is intended to do, and how each team thought it best to go about their work. Not only were there problems to be sorted out between each time, but there were also moments where our team and the client had to sit down and talk about what was expected of the project. Not only did each of us have different interpretations of what was to be expected, but we each had different ways that we thought best to go about it. Resolving each of these was a major component of getting our project to work properly.

There was also the issue of understanding what the project was about, even within our own team. Each of our team members has a different background, and those backgrounds could interfere with understanding. For example, two of our team members were electrical engineers and had difficulty understanding the fine details of software implementation and testing, whereas the software engineer on our team did not understand the electrical component of the project. This hindered progress between teams, as well, since there were certain details that escaped each team due to not fully understanding the scope of the work.

On a whole, we were able to resolve these problems by explaining the details of the implementation to other members of our team and between our teams to the best of our ability,

as well as providing extra assistance to members of our team who needed to know each component not only for themselves, but also for communication with the other team and with the client. This worked well on a whole, but it still proved difficult sometimes to gain a full understanding of the project if one of our team members needed to know parts of our project that pertained to the other team's success. In the future, a good way to circumvent this problem would include a thorough assessment of each member's understanding of the work as well as determining which components of the project need to be understood by other members versus which components only need to be glossed over. This would allow us to streamline our communication process and provide assistance to people who require extra understanding early so that our project is not hindered by these issues in the future. This lesson applies both to our own team as well as communication and understanding between each of our teams.

iv. Constant communication

On a whole, we feel that our teams did very well with keeping in constant communication throughout both semesters. However, the importance of constant communication is not lost, and is one of the most important lessons that we learned.

In order to keep both of our teams informed on the issues that the project faced as well as progress that was made, we set up weekly meetings with each other. In our first semester, we would meet with the hardware team every week after class to inform each other of progress, issues, etc. By doing this, we were able to keep a constant line of communication open and had a consistent time that ensured an exchange of information would occur.

Moving into the second semester, we scheduled a weekly meeting that included both teams as well as our adviser /

client. This weekly meeting proved to be incredibly useful, since it guaranteed us not only time to communicate with the other team, but also guaranteed that there would be time to speak with our client on the issues that we found and correct them early in the design process. These meetings prevented us from dealing with a lot of hardships since we were able to bring up problems while they were small, as well as review our requirements and work towards viable solutions together.

If we could change one thing between what we had done and what we would do in the future, we would hold the typing of meetings that occurred during the second semester with both teams and our client in the first semester. Although we did have a meeting set with the other team in the first semester, it was fairly informal and did not include our client. Instead, each team met with the client separately. It could have benefitted us to hold the large group meetings to begin with, rather than beginning to do it halfway through the project.

- b. Although it sounds rather intuitive, a big lesson that we learned was beginning our project early. While we did not slack off in working on our project and thinking about the problems that we would face, we still found ourselves a bit rushed at the end to complete everything that we needed to do. Although it was difficult to see the ways that we could improve at the time, looking back now, we can see the main reasons to improve our efficiency that would allow us to improve on our work.

It can be difficult to pinpoint exactly how we would change this in the future. The most efficient path seems to become most visible after all information is laid out, and at the beginning of a project, we do not have all of the information that we need to make the best decisions. However, the lessons that we learned about working efficiently and how they impact our final project can be explored in

full detail and kept in mind for future projects.

i. Malfunctions through the project

For more detail on this, see part C. However, the fact that anything can go wrong is important information to keep in mind when doing the project; the concept of expecting the unexpected is important in any project. Within our project, a number of things went wrong that hindered our progress. The things that went wrong ranged from code not working the way that we expected it to, misunderstanding the way that libraries work, etc. to things that were out of our control, such as hardware breaking.

It is difficult to cover for every possible scenario of possible malfunctions in a system. However, certain malfunctions can be foreseen. Hardware breaks and nonfunctioning code are fairly frequent scenarios; ergo, it could help in the future to have a plan to not only deal with these problems, but have an understanding of what can be worked on and how to make progress in the meantime while these problems are being fixed.

What we learned from this lesson was that, while we cannot foresee all malfunctions, predicting and understanding possibilities based on past experience and the hardware and software presented to us is necessary. Having a plan to not only deal with these problems but make progress in the meantime is vital to the smooth continuation of the project.

ii. Exploring our options

A major benefit of starting the project early is the ability to explore different options available to us in terms of both hardware and software while still allowing us enough time to understand what is available to us and utilizing those options to the fullest.

For 3D printing, there are numerous hardware and software components available to us that assist us in different ways. Part of the project was to explore all of these different options and decide which ones would best suit our needs, and which ones would provide us the best functionality and best extendibility. Even though 3D printers are a fairly new field, there are a number of viable options available.

Part of the benefit of starting early is the ability to explore all of these options without worry for whether there will be enough time to fully learn them and utilize them to their full potential. In terms of both the GUI and the back-end printer work, there were numerous changes (as detailed in Appendix II). Each time we would have to make a switch to a different type of software, the time that we had spent learning the previous software's capabilities and functionality was lost.

While it seems inevitable that learning more about the available software as well as the needs of our client and the limits of our hardware would provoke a change in the software that we are using, there is still potential for wasted time and effort that, ideally, could be avoided with careful planning and consideration.

Much like hardware and software malfunctions, it is difficult to discern all of the options available to us. It is unlikely that we would be able to find the ideal hardware and software on our first try. However, we would like to think that the wasted time could be minimized through careful steps and planning. A possible way to circumvent wasted time is to do Internet research on the software available to us and compare it with the requirements that we have to determine if it meets all of our needs. Additionally, we could ask other people who have had to build and work with similar software what their experiences and recommendations are, perhaps through an

Internet forum or other public means. This bit of effort could go a long way in preventing wasted time and allow us more time to start on the meat of our project early and get more done.

The lesson that we can take away from this is that, while we cannot necessarily predict all of the software and hardware that is available to us, there are certain steps that we can take to help us minimize wasted time and get started sooner. In the future, we could ask around and do more thorough and interpersonal research to better understand everything that is available to us.

iii. Less rush

One of the most important takeaways from this project was that, with good planning and a full understanding of what is available to us and how to predict problems, it saves us time and trouble in the long run.

Projects are much easier to manage if it is done like a marathon, pacing out effort over a long run, rather than like a sprint with a burst of effort at the end. While we attempted to make our project work like a marathon, the aforementioned problems that we had sometimes made it feel like a sprint in the end.

In order to make the project feel less rushed and spend more quality time on it, we learned a lot of lessons on how to best manage our time, work with our project, and predict for disaster. In the future, we would implement each of these plans to most effectively work towards improving the quality of the time spent on our project as well as implement meticulous strategy on how to deal with problems as they arise.

- c. Throughout the course of a project, anything can go wrong. Although we briefly discussed some things that went wrong in section B, the full impact and scope of all of the disaster that can hit the project is very important and will be discussed further.

It is to be expected during a project that hardware or software can malfunction, and these malfunctions should not only be considered before they happen, but need to be dealt with in a timely manner when they arise. There are multiple types of malfunctions that can occur. Exploring them and determining the best route to take to either fix or prevent them is vital to the life and efficiency of a project.

Although the hardware was beyond the scope of our team's project, the hardware and software components of each of our teams is closely married. If there are hardware malfunctions, it can prove difficult for our team to work and test what we have. The broken components that we dealt with over the course of the project ranged from the technically testable but still unsavory, such as safety components on the printer being broken, to problems that outright stopped us from using the printer, such as fried electrical components. Dealing with these problems can hinder the project greatly, as it prevents us from being able to see whether our hardware interfaces with the software. In addition, we must wait for the components to be fixed.

Broken components were not limited to just the hardware, however. Our software had its own difficulties, ranging from software stacks not working as expected and being unable to interface with the printer to outright syntax errors and being unable to find paths that the printer needs to utilize all of the software together and communicate with the printer. Fixing these problems, while part in parcel of designing software, slowed down our project and forced us to troubleshoot the components that needed to be fixed rather than being able to work on new portions of the project.

We were able to work through these problems by taking time to troubleshoot and having patience for the repair of parts of the project that were outside the scope of our work. In the future, we may also find it prudent to do the best work that we can without the components and test what we have when the components are working again, as well as explore all of our options for what we can utilize to minimize difficulties in terms of software that we choose to work with.