ILR03

10/23/15

TEAM F: ADD_IN

Dan Berman
Astha Prasad
Ihsane Debbache
Nikhil Baheti

Individual Progress

Since the last ILR I have focused on the following two areas:

1. **Updating project plan for Systems Engineering Presentation**
   On Tuesday, September 27th, our team gave a presentation to the systems engineering class detailing changes and developments made to our project since the conceptual design review. As part of the preparation for this presentation, I went through our software work breakdown structure, functional architecture, and fall validation experiment and updated them to reflect our current intentions based on lessons learned over the last three weeks. Specifically, since learning that we can completely control Slic3r through command line tools, and thus through MATLAB, we have adjusted our functional architecture to use MATLAB post processing to handle insertion layer selection and generation of 4 DOF G-Code commands (as shown in Figure 1). This also produced significant changes in our WBS since we have now pushed implementation of layer selection and path planning within the Slic3r software to the spring semester, and are focusing only on a MATLAB implementation this semester to get the system up and running as quickly as possible.

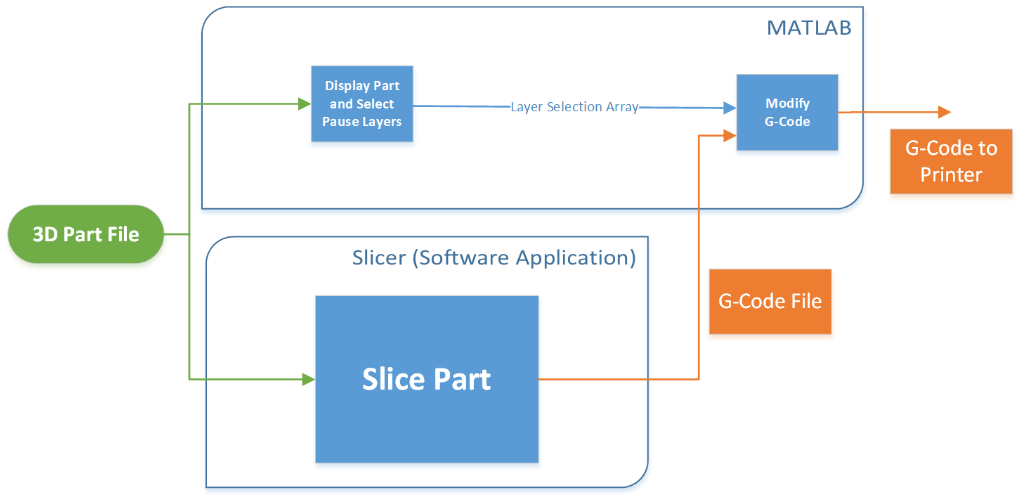

Figure 1: Slicer functional architecture of the Slicer software showing components which will be implemented in MATLAB instead of being incorporated into the original Slic3r package.

2. **Creating the conceptual design of our PCB board**

   To develop the conceptual design for our PCB I determined and analyzed the key components and circuits which would most significantly affect the schematic and board design. Specifically, I chose a communication protocol to use, picked an ADC, and determined the ADC settings and external resistor sizes needed to optimize the circuit for measurement of the 3D printer's nozzle temperature. Since the circuit board will be digitizing sensor data to send across a slip ring, I placed a significant weight on minimizing the number of channels required to communicate with the main RAMBo board. Originally, I had hoped to used a Maxim DS2450 IC, which provides a 4 channel ADC using Maxim's 'one-wire' interface. This protocol multiplexes power and serial communication into a single line, thus only requiring two wires (power/signal, ground) to interface the IC. Unfortunately, Maxim has discontinued manufacturing this IC and thus I designed for an I^2C interface (requiring 4 lines: power, ground, clock, data). This is still not an ideal solution since I^2C is not an especially robust network, but the preferred alternatives (CAN or RS-485) would have required significant changes to the RAMBo board and printer firmware. Fortunately, our required data rate is extremely low (the thermal system has a long response time) so by using a combination of electrical filtering and a software algorithm to discard corrupted packets we hope to still accurately control the nozzle temperature.

To analyze the ADC and determine the best gain settings and external resistor values I created an excel sheet which mapped a temperature at the thermistor to a numeric value and sensitivity as read by the ADC. The template I developed included effects from ADC impedance and thermistor non-linearity as approximated by the Steinhart-hart equation. The results allowed to me chose a gain setting and resistor value which produced good sensitivity (~0.4°C resolution) and avoided saturation across our nozzles operating range (180°C - 220°C), as shown in Figure 2.

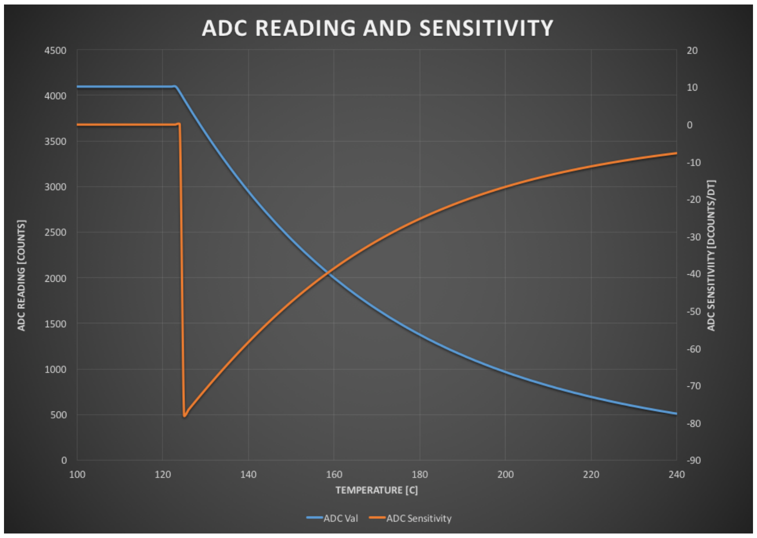

Figure 2: ADC Output Value and Sensitivity as a function of thermistor temperature for a PGA setting of 4 and 10K ohm fixed resistance. Note ADC saturation below ~128°C

With these components and values determined, the only design work remaining is the selection of passive components and a suitable voltage regulator. An analysis was performed to determine the required parameters of the voltage regulator, but the exact model will be selected while developing the schematic and board layout.

Challenges

As mentioned last week, the Slic3r software has been installed from source, but both Astha and I have run into an issue installing the GUI module. This module fails due to the inability to install the ‘Alien::Wx’ package (terminal output shown in Figure 3), which provides the porting necessary to install Wx, a GUI development tool Slic3r is built upon. I spent this week extensively researching the topic, reviewing log files, and trying to determine the exact reason why Alien::Wx fails to install. Without prior experience with Perl libraries this has been extremely difficult and largely unsuccessful. I have found a reports of Alien::Wx failing to work with recent operating system updates. The Alien::Wx module’s original author seems to have stopped providing frequent updates and patches to the software, and thus it is possible that Alien::Wx will soon be no longer supported. Fortunately, having the source code to the GUI portion of Slic3r in the fall semester is not essential to our project. I have emailed Slic3r’s original author for suggestions, and will continue to devote 1-2 hours per week to trying to resolve this issue.

```
OpenGL is up to date. (0.6704)
Socket is up to date. (2.020)
--> Working on Wx
Fetching http://www.cpan.org/authors/id/M/MD/MDOOTSON/Wx-0.9927.tar.gz ... OK
==> Found dependencies: Alien::wxWidgets
--> Working on Alien::wxWidgets
Fetching http://www.cpan.org/authors/id/M/MD/MDOOTSON/Alien-wxWidgets-0.67.tar.gz ... OK
Configuring Alien-wxWidgets-0.67 ... OK
Building Alien-wxWidgets-0.67 ... FAIL
! Installing Alien::wxWidgets failed. See /Users/Dan/.cpanm/work/1446071977.9991/build.log for details.
Retry with --force to force install it.
! Installing the dependencies failed: Module 'Alien::wxWidgets' is not installed
! Bailing out the installation for Wx-0.9927.
```

Figure 3: Terminal output of Alien:Wx install failure when attempting to install Slic3r from source.

Teamwork

Astha Prasad: Astha made significant improvements to the initial layer selection MATLAB script we produced last week. She has also identified and solved a quirk of Slic3r, which was preventing us from loading configuration files when invoking Slic3r from command line. This quirk was preventing us from developing the G-Code needed to produce real parts since Slic3r was forced to using default settings which are not compatible with our printer. After identifying the quirk, she developed a robust solution to it within our MATLAB script, thereby enabling us to create a single file with all of the appropriate Slic3r settings for our printer which is loaded at runtime.

Nikhil Baheti: Worked extensively on the CAD design of the nozzle. He has already iterated the design a few times and converged to a result which Ihsane and I will begin machining next week. In addition to this, Nikhil put significant time into development of the systems engineering presentation. Specifically, he created a presentation outline for the entire team to work from, produced all of the charts and figures for the risk management section and critical path sections, and compiled all of the final slides.

Ihsane Debbache: Did extensive research to determine the design requirements for the nozzle. He worked closely with Nikhil, developing and passing him the requirements, who then converted them into a CAD drawing. Ihsane also handled procurement, ordering many nozzles, barrels, and heat blocks, and raw materials necessary for building and testing nozzle designs.

Future Plans

In the next two weeks my primary technical focus will be working with Astha to develop the path planning algorithm to control the rotational axis being added to the printer. The path planning algorithm will be implemented in MATLAB as a post processing script for the G-Code output of Slic3r. In addition to the G-Code output, the path planning algorithm will also need to receive position and possibly dimensional information of the COTS items to be added. Determining exactly what this format is will be one of the firsts tasks I begin on, but may be as simple as a XYZ coordinate and radius to form an approximate bounding cylinder which the aft portion of the nozzle must avoid. Currently my plan for the algorithm is that it will interpret the G-Code commands output from Slic3r to control the rotation axis so that the nozzle always has the same angular orientation relative to it’s trajectory. For most orientations this will yield two solutions symmetric across the trajectory, which will be chosen between by maximizing the normal distance between the bounding box of the nearest COTS item and the cold end of the extruder nozzle.

One potential issue with this plan is the situation when printing adjacent horizontal lines where at the end of each line the nozzle may be required to rotate extremely fast to produce a small bend radius to begin the next line. A special case may need to be added to the algorithm to address this situation based on the mechanical limitations of the rotation joint. More information regarding the likelihood of this posing an issue and necessary corrective steps will emerge in the next week as Ishane and Nikhil begin design work on the rotary stage.

In addition to my work on the path planning algorithm, in the coming week I will also be helping Ihsane to machine the first iteration of the nozzle design and working with the entire team to develop our PDR presentation.