Progress Review 4

Nikhil Baheti

Team F: ADD_IN

Teammates: Ihsane Debbache, Dan Berman and Astha Prasad

ILR04

November 13th, 2015

1 Individual Progress

This week I had to work on modifying the CAD file of the heat block easier to machine. Another objective for this week was to understand the firmware and its general flow of code. After understanding the firmware, the next task was to control the R-axis stepper motor using the Extruder1 port on the Rambo board. For this progress review, I also worked on the layout design, and on the generation of the manufacture files and BOM for the PCB assignment.

1.1 Solidworks Design of Aluminum Heater Block:

I had to modify the aluminum heater block for easier machining. Dan suggested that the entire block be level on one side to 45 degrees as it would make machining easier and less confusing. So, I made modifications to the design and the corresponding drawing is shown in Figure 1.1.

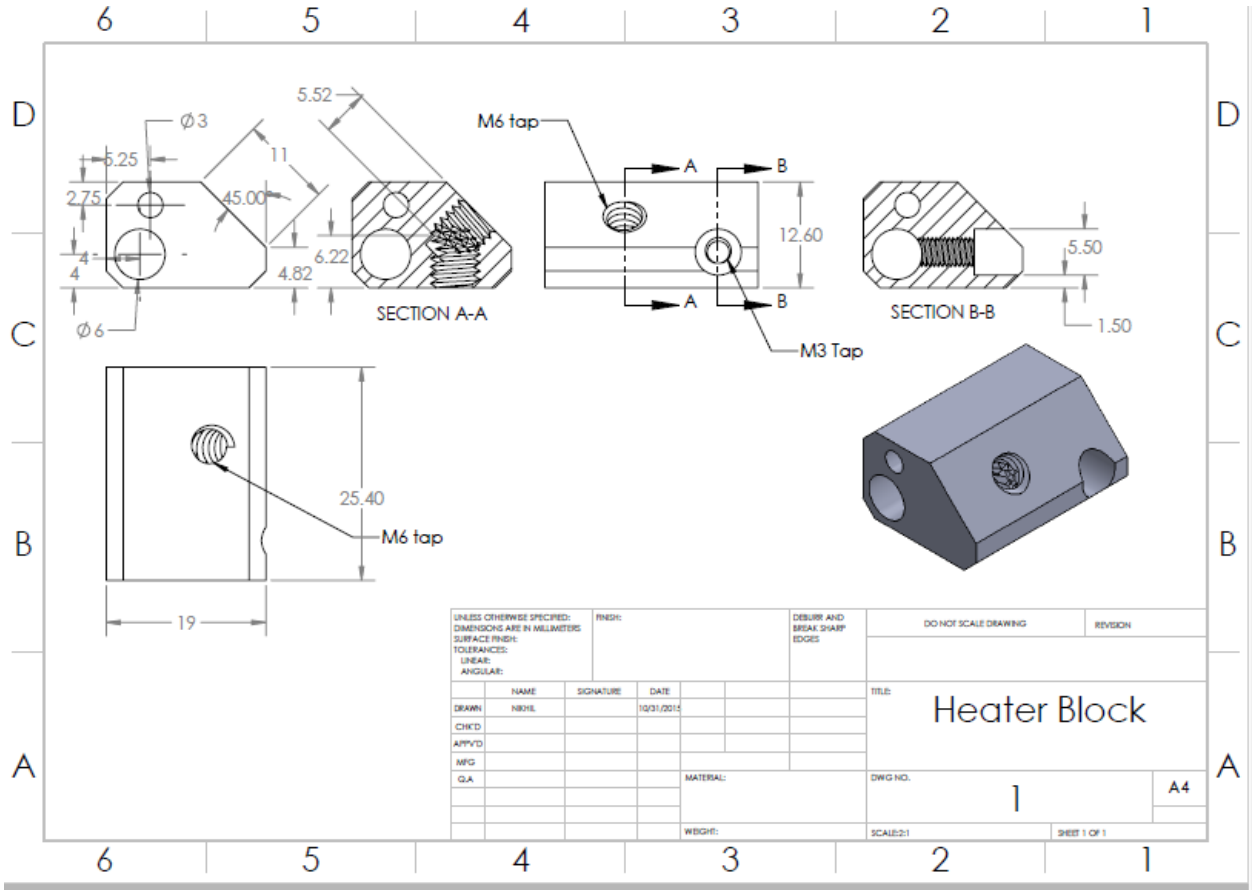

Figure 1.1: Final drawing of the heat block

1.2 Understanding the Firmware:

The Marlin firmware is easy to understand. But it uses many libraries and also has many different files for controlling different parts of the printer. Thus, I had to record information to avoid going back and forth between the files. I have started using Visio as a tool. All files are mapped on an overview page. While mapping these files a lot of files I discovered were needed used for the printer to interface different modules and of the printer with Rambo. Files that interface SDcards, LCD screen, watchdog timers, servo motors and Rambo EEPROM programming do not need modifications. However, the crux of 3D printing exists in the following files:

1. **Marlin_main.cpp:** This file contains the setup and loop functions. It parses the Gcode commands and invokes calls to the motion control functions of the axes.
2. **Configuration.h:** This file defines initializations for board type, power supply, no of extruders, thermistor settings, mechanical settings, and servo support. Rambo is identified by the "MOTHERBOARD" variable set to 301.
3. **Pins.h:** This file defines the interface pins for various controllers.
4. **Marlin.h:** This file defines USB communication functions, motor enable control functions and PWM variable definition.
5. **Fastio.h:** This file defines the macros for the IO ports read/write functions and register definitions for the IO ports.
6. **Stepper.cpp (Header File: Stepper.h):** This file controls the stepper motor motion by writing onto the control pins of the stepper motor.
7. **Temperature.cpp (Header File: Temperature.h):** This file controls the PID variables that control the temperature for both extruders.
8. **Motor_Control.cpp (Header File: Motor_Control.h):** This file defines the next stepper motor positions by synchronizing all the stepper motor motions.
9. **Planner.cpp (Header File: Planner.h):** This file defines the profile along which the stepper motors must traverse to complete the Gcode execution. It may be a line or an arc.
10. **Speed_Lookuptables.h:** This file contains the speeds for the stepper motors for various accelerations in the form of a look up table.
11. **Thermistor_tables.h:** This file contains the thermistor readings look up table. It maps the ADC digital value to the corresponding temperatures.

Also, for a few files the sub functions have been mapped in the visio chart and are cross linked with their parent and child calls. The visio chart will be updated as and when required to make sure that the person working with the firmware can easily make modifications. Figure 1.2 shows the visio layout for "Marlin_main.cpp."

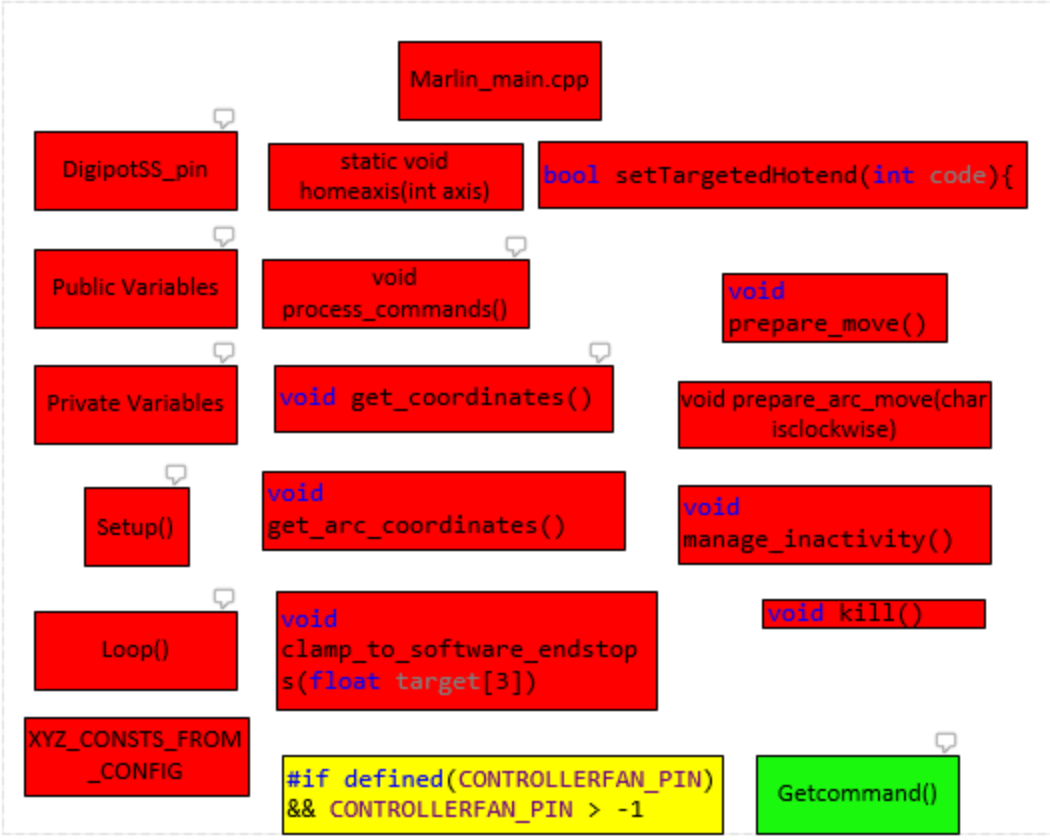

Figure 1.2: Marlin_main.cpp functions. Red-> Function must be modified; Green-> No need to modify function; Yellow-> Function may have to be modified

1.3 R-axis Stepper Motor Control:

The R-axis stepper motor is controlled using the extruder1 port on the Rambo board and its corresponding control pins. The control code for the R-axis stepper motor uses the libraries that were created during the sensors and motors control task for stepper motor control. A new Gcode "G5" was defined in which the control_stepper() function of the stepper motor control code is invoked. However, the stepper motor currently moves only a fixed pre-assigned distance. In future the control of the stepper motor will be incremental modified to reach the final synchronized control with motion planning.

1.4 PCB Layout design, BOM and Manufacture files:

I also worked on the PCB task to make the layout, BOM and manufacture files for the circuit which will be used to digitize the thermistor data. The layout of the PCB has been optimized to reduce the area occupied by the PCB while simultaneously not be wider than 0.5" so that it can be mounted on the insulator PEEK. The ADC sensing lines have been traced to ensure that the data experiences minimum capacitive interferences due to the other nearby data or ground lines.

2 Challenges

The challenges faced during the term of this progress review are as follows:

1. **Arduino IDE:** The Arduino IDE for tracing variable declaration, references and initiations is not user-friendly. Thus, I had a hard time tracing variables across files. Therefore, the need for the visio chart is necessary. Finally, I moved to Visual Studio as an IDE. But since the Arduino inherent functions cannot be mapped by the Visual Studio IDE, I have to constantly move back and forth between these IDEs and take care of changes.
2. **PCB Layout Deisgn:** The PCB layout had to be optimized for area and it took a lot of time to achieve this. First few tries were scrapped as my approach was to place all components and then route them manually which wasn't probably the best way to do it. Finally, I did a batch wise routing wherein I placed and routed sections of the PCB and then moved to the next section.
3. **PCB manufacture files generation:** The DFM free testing software produced slik screen errors despite the fact that the silk screen for all components was 0.05" above the required minimum of 0.005".

3 Teamwork

- **Astha:** She worked on displaying the stl code and insertion layer on Matlab. She has also taken over as the project manager and is doing a great job.
- **Dan:** He worked on the path planning algorithm. His code can plot the motion of the nozzle, generate the Gcode with the R-axis co-ordinates and show the direction of the R-axis that the nozzle must point to while printing. He also machined the heat block.
- **Ihsane:** He was responsible for testing the machined heat block. He also worked on the CAD file designing for the rotary stage assembly. He found a hollow stepper motor which we will use for the project. He was also responsible for selecting the slip ring.

4 Plan

For the next progress review, I plan to work on the firmware and control the R-axis stepper motor given a relative position to extrude. This will be independent and not in synchronization with the other axes of the printer. However, the team priority for the coming progress review is making the nozzle print filament uniformly. Thus, I will be helping Ihsane whenever he needs help with the heat block design iteration.