Progress Review 8

Nikhil Baheti

Team F: ADD_IN

Teammates: Ihsane Debbache, Dan Berman and Astha Prasad

ILR07

January 28th, 2016

1 Individual Progress

This fortnight, I had to work on completing the firmware for the 3D printer. T he control of the r-axis stepper motor through micro stepping, similar to other axes was one of the major tasks. This includes solving the hardware problem on the E1 port. I also had to modify the other G-codes where the r-axis variables are altered to ensure that the system is consistent across all axes.

1.1 Hardware working of E1 port

To solve the problem of E1 port, I first tried microstepping on other ports. After having learnt how this works I tried to do the same with the E1 port. However, it did not work. Then I tried changing the pins of y-axis to control port E1. However, this did not work too. To eliminate any dependence on the firmware, the microcontroller was programmed with the original firmware and then the y-axis control pins were used to control the E1 port. Failing to drive the port the problem was narrowed to the hardware. Hence, the boards have been changed and we are currently using the spare board that we bought the previous fall semester.

1.2 Firmware working of E1 port

After changing the board the E1 port did not respond to the modified firmware and thus the following steppers were used to debug the problem:

- Control the E1 port using the original firmware. This did work which concludes that there is no hardware problem
- Use R-axis variables to control Y-axis port. This did work which concludes that the variables that are defined are performing the required operations.
- Comment the y-axis pins since there is a dual definition of pins (r-axis and y-axis) controlling the same ports. This produced compilation errors as there are a lot of places where these variables are used in the code and there is no time efficient way of eliminating these variables.
- Use the compilation errors to check if any initializations in r-axis have not been performed when compared to the y-axis. This did help point out a couple of changes but there weren’t causing the problem. This was reconfirmed by removing similar initializations for y-axis and checking if the y-port can still be controlled.
- The only logical reason that I can think of is that the uninitialized E1 variables are resetting the port and deactivating it. However, this part of the code is not traceable.
- So, I set ADD_IN variable to 0 to make sure any changes in the firmware are removed. Doing this still did not activate the E1 port. So I used the winmerge tool to check the differences between the original Marlin firmware and the modified

firmware. This resulted in the firmware controlling the E1 port without ADD_IN features.

- This narrows down the variables to the ones that were changed in this process.
- Now the approach is to check where each of these variables change and try to find the solution to the problem.

Figure 1 shows the flowchart of the approach used for debugging.

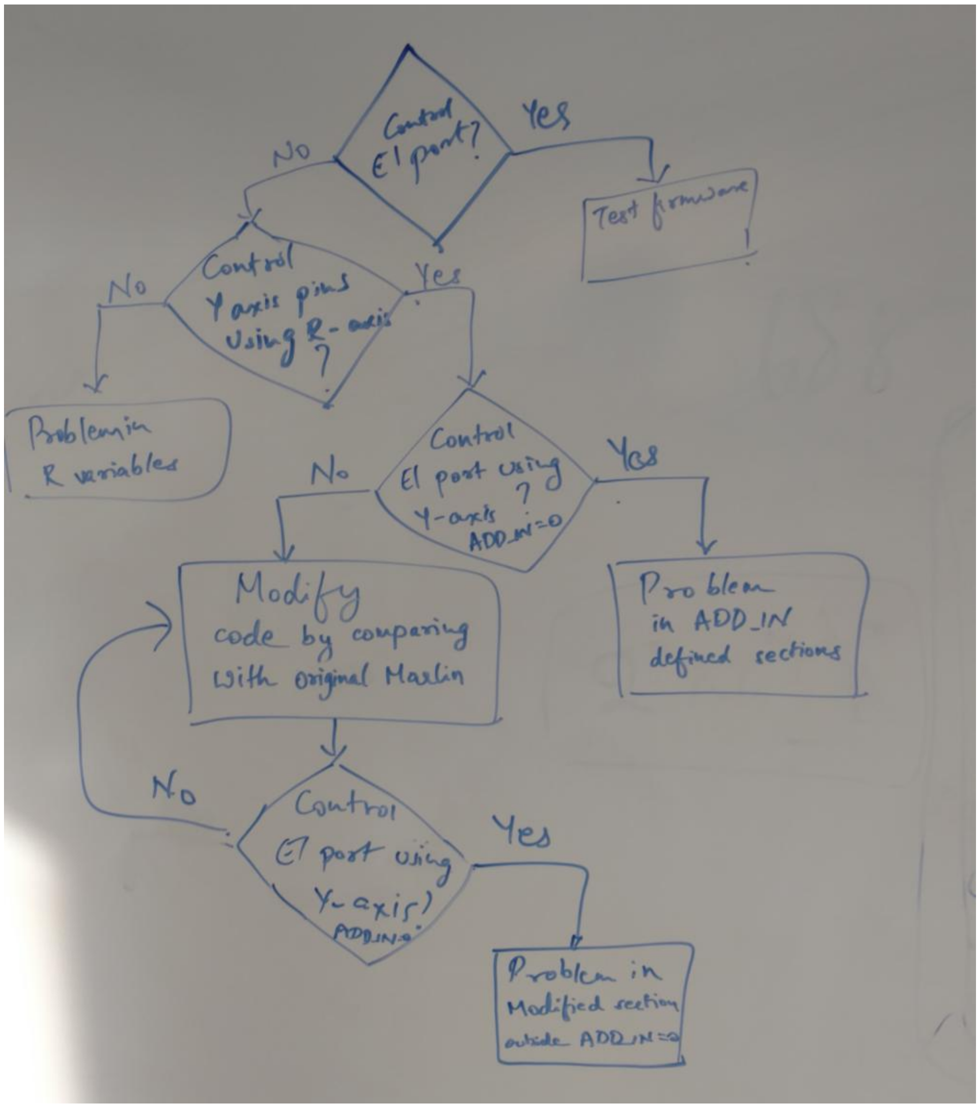
Control E1 port?
Yes
No
Test firmware
Control Y axis pins Using R-axis ?
No
Yes
Problem in R variables
Control E1 port using Y-axis ? ADD_IN=0
No
Yes
Modify code by comparing with original Marlin
Problem in ADD_IN defined sections
No
Control (E1 port using Y-axis) ADD_IN=0
Yes
Problem in Modified section outside ADD_IN=0

Figure 1: Flowchart of the debugging process

1.3 Modifying different G-codes

Various G-codes use the x, y and z axis variables to update and use them. Like G92 is used to set reference points. I had modified these codes to work for r-axis. However, the E1 port not working has forced the development to previous versions as it is easier to narrow down the changes. Thus this development must be done again but it will be quicker using the winmerge tool.

2 Challenges

The challenges faced during the term of this progress review which are quite consistent with the previous problems and are explained 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.
2. **Debugging:** The firmware could not include break points to monitor the variable which resulted in a lot of time waiting to write the code even for a small change in the code. Also, I had to use print statements echoed to the serial terminal to debug sections of the code. The essential part of the code exists in the timer interrupts where the stepper motor is controlled. This is called every few milliseconds and thus printing in this interrupt is not useful as it overloads the buffer on the serial communication port causing large outputs which do not make sense. The current approach to tackle this problem is using a static variable that ensures prints only every few seconds.

3 Teamwork

- **Astha:** She worked on making a GUI in MATLAB that can import slicer setting and visualize the various steps of the process. This will help us accelerate the testing phase as the GUI can be used by anyone not familiar with slicer.
- **Dan:** He also worked on the GUI in MATLAB that can import slicer setting and visualize the various steps of the process. Apart from this he was responsible for machining the new nozzle designs and he also designed one of the new models.
- **Ihsane:** He was responsible for testing the machined nozzles. He also had designed a nozzle and the hollow tube that passes between the stepper motor, slip ring and the heat block.
- **Team collaboration:** There was no major component of team collaboration as most of the work was divided, but we did meet regularly to check progress. We also met to try and integrate the subsystems but since the subsystems had issues associated with them we could not integrate the system.

4 Plan

For the next progress review, I plan to work on the firmware and modify the code for the printer to use all G-codes and solve the bug in the firmware that is restricting the control of the E1 port.