- **Progress Review 7**

Nikhil Baheti

Team F: ADD_IN

Teammates: Ihsane Debbache, Dan Berman and Astha Prasad

ILR06

January 28th, 2016

# 1 Individual Progress

This fortnight, I had to work on modifying the firmware for the 3D printer. This included completing the control of the R-Axis stepper motor through micro stepping, similar to other axes control. I also had to modify the path planner for controlling the R-axis synchronous to the other axes. Also, to reduce time to modify the code I explored other CNC codes and 3D printer codes which use 5-axis controls but I failed to find one.

## 1.1 5-Axis Control Codxe Search

I thoroughly searched for the available firmware for 5-axis CNCs and other 3D printers firmware codes like R-360 and dual extruders. However, these cannot be used for the following reasons:

- The firmware is not RAMBo board compatible. These include CNCs' firmware.
- The firmware has an extra axis but the control is the extra axis is not similar to the R-axis. These include Marlin (our current firmware), R-360 (just modifies the Y-axis to control R-axis) and dual-extruder printers.

Having failed in the search for a firmware that we could just modify to use I moved onto modifying the code one line at a time starting with stepper motor control.

## 1.2 R-axis Stepper Motor Control

As mentioned before we are controlling the R-axis using the E1 port. However, the E1 port has been implemented as an extended control of E0 axis and therefore, the entire code for R-axis had to be written down. The following was my approach to modify the code:

- Check the entire file for y-axis stepper motor control code and duplicate the variables, functions and lines of code for R-axis. The y-axis was used because it has no special functions. The x-axis may be used as dual carriage, the z-axis is only used increasing the height per layer and e-axis has a completely different way of controlling itself.
- While copying the code I had to make sure that I do not copy functions that the R-axis will not use like encoder reading and end stop checking.
- Modify the variables in configuration and other files to ensure the consistency and avoid compilation errors.
- Use as ADD_IN flag to ensure that it can be set to false so that the Marlin can be used in the default configuration.

## 1.3 Planning Modifications:

After modifying the stepper motor control file, I decided to approach the problem by modifying only one G-code command. So, I only looked at functions associated with G-code "G0" and "G1" which had files like planner.cpp. Modifying the planning algorithm was the most challenging task. The approach taken is as explained below:

- Perform a line by line manual interpretation of the code starting from "Marlin_main.cpp."
- Understand the flow of the algorithm in "planner.cpp."
- Modify the variables in configuration and other files to ensure the consistency and avoid compilation errors.
- Use as ADD_IN flag to ensure that it can be set to false so that the Marlin can be used in the default configuration.

## 1.4 Integration and Testing

After the above modification I had to integrate all the modifications and test the modifications. The procedure followed is explained below:

- Modify the setup and initialization functions
- Modify variables that are initializes by default configuration codes
- Write the firmware to the microcontroller and debug the code. Iterate this process till the firmware works.

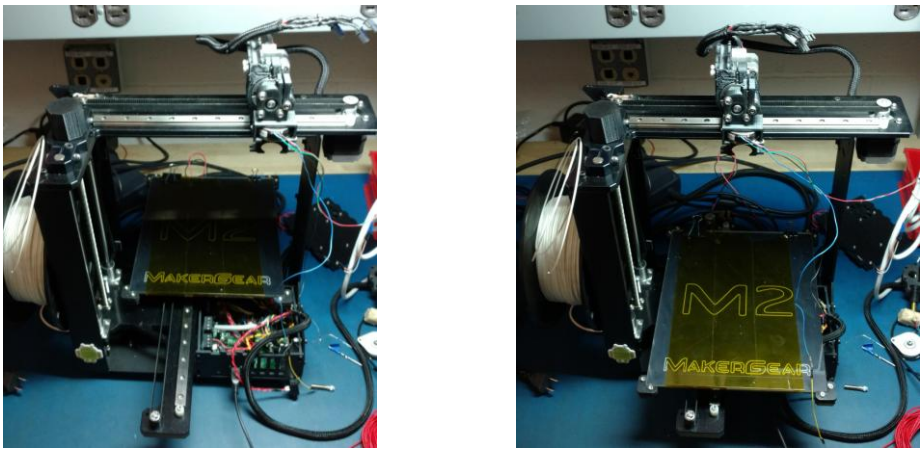


Figure 1: (left) shows the Bed and X axis motor status before G0 X-20 C20 execution; (right) shows the status after G-code execution

Figure 1 shows the printer status before and after execution of a synchronous G-code. Some debug techniques used were:

- Use the serial communication functions to print on to pronterface interface.
- Keep track of variables add interlinking functions like stepper_control and planner.
- Replace E1 motor control pins to working port motor pins to eliminate any hardware issues.

## 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. I also tried using eclipse as an IDE but the arduino libraries are not compatible with it.
2. **Marlin Coding:** Marlin is an open source code. Thus there are a lot of modifications where the variables are not consistent. This resulted tedious task of checking every for loop and axes variables to ensure that they are consistent. For example, for loops containing axes variables should vary based on the variable "NUM_AXIS" which defines the number of axes. However, there are a lot of instances where the code uses the constant 4 instead.
3. **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.

## 3 Teamwork

- **Astha:** She worked on modifying the path planning algorithm to ensure it may work in all corner cases. This was mainly backend modifications to ensure that the program is structured.
- **Dan:** He also worked on modifying the path planning algorithm to ensure it may work in all corner cases. These were mainly modifications to ensure that the planner is planning the path with consistency.
- **Ihsane:** He was responsible for assembling the stepper motor and the slip ring. He also worked on the next nozzle iteration design.
- **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 brainstormed ideas for the next nozzle design.

# 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 start testing by printing objects. I will also, solve the hardware problem on the RAMBo board, as it is now failing to control E1 axis using micro stepping.