Lab 3

Justyn Harriman

November 17, 2014

1 Download and build firmware for you PrintrBot

I was not able to produce a matching HEX file. See my unanswered post on Piazza for details of what I tried: `https://piazza.com/class/i0tylwj4fgr2ia?cid=55`

2 Read the source code for the firmware and answer the following questions

2.a Explain what algorithm is used to manage the heater extruder.

The firmware uses a PID controller in order to manage the amount of voltage to send to the heaters. Each time manage_heater is called, the firmware calculates how different the current temperature is from the targeted temperature; this difference is called the "error". Then the firmware calculates the PID output using the following algorithm:

Initially, set D to a random value? (The code never assigns the unintialized $dTerm$, so I assume they are hoping it is random?)
for all calls to manage_heater **do**
Set P = The K_p tuning constant multiplied by the current amount of error
Set I = The K_i tuning constant multiplied by the amount of error since the last reset (which could be when we started heating the extruder)
Set D = (The K_d tuning constant multiplied by the difference of the current temperature and the previous temperature) * a smoothing constant K_2 + (a smoothing constant K_2 * D.
end for

2.b Explain how Marlin implements arcs (G-codes G2 & G3) motion_control.c

G-codes G2 & G3 can take the following arguments X(x.x) Y(x.x) I(x.x) J(x.x) E(x.x), where (x.x) is an arbitrary floating point number.

- (X,Y) are the coordinates for the destination point relative to the current position of the print head
- (I,J) are the coordinates for the center of the arc (the position that the line will remain in a fixed position to) relative to the current position.
- E is the amount of filament that will be extruded over the arc.

From these inputs, Marlin first calculates: the center position of the arc, the z-axis travel over the course of the arc, the extruder length to extrude over the arc, and the total angular distance between the current position and the target position through the center of the arc. Then, Marlin creates an 'arc' by splitting the angular distance into a set of linear 'segments' that are all a fixed distance. It prints these linear segments in sequence, interpolating the line across the angular distance, until the destination point is reached. Marlin also accounts for floating point errors over many iterations of this rotation by periodically refreshing the value of the in-memory axis with more accurate values computed from the beginning of the arc.

2.c Explain how the steppers are controlled using "blocks"

Planner.cpp has a block_buffer of "blocks" which contain fields for the number of steps a stepper motor should take during specific phases of execution. In "plan_buffer_line", Marlin takes a target destination as input and plans a line between the current position and the target position. To do this, it creates a "block" structure which contains a set of stepper motor steps to take in order to accelerate to the maximum velocity the block will achieve, the number of steps that the motor will 'cruise' at the maximum speed, and the steps required to decelerate to a minimum speed. After a new block is created, the planner adds the block to the block_buffer and then recalculates the acceleration and deceleration steps in order to mesh all the blocks in the buffer together smoothly. Some blocks do not need to be recalculated, and these ones are passed over by the recalculation. The planner smooths out these motions, in order to minimize jerks in motion by the stepper motors. By recalculating the number of acceleration and deceleration steps

2.d Give an example of how speed_lookuptable.h is used in the firmware.

It's used in line 267 of stepper.cpp and is used there in order to generate the timer that clocks the steps made by the stepper motor. The lookup table changes based on the CPU and the, acceleration and deceleration step rate calculated for a stepper block.

2.e What is the watchdog timer used for?

The watchdog timer is used to make sure that firmware doesn't block for longer than 1 second. If there is a function block for longer than 1 second, we can assume the printer encountered an issue and requires reset.

2.f Give some examples of what the planner is used for.

The planner is used to plan motion between the current position and a new position using "plan_buffer_line". It schedules this planned change in motion by creating a "block" that is then added to the block buffer.

2.g Where is assembly language used in the firmware? And why?

For an example, line 127 of stepper.cpp a function called "MultiU16X8toH16(intRes, charIn1, intIn2)" is defined that returns a function in assembly language. This function returns an int that contains the two high bytes of the result of multiplying charln1 and charln2. Assembly language is used in this function because hard coding of the assembly language is much faster than letting the compiler do the same operation.

3 Build a block control flow diagram that covers the general functions of the Marlin firmware and how they call each other.

The block diagram image is in the file in this zip called "Digifab Control Flow Diagram.png"

4 Writing G-Code without Slic3r

Write by hand or via a program you write a simple g-code file to print a 2 cm sphere and run this g-code on your printer. Note: You may borrow the prelude and post-print headers and footers from g-code that has been generate by slic3r.

Compare your hand/program generated g-code to the g-code produced by the normal tool chain for pring a 2 cm sphere (i.e. openSCAD! Slic3R)

I have included code to generate hemispheres in pygcode folder of this zip. The README contains instructions for running my python script. The script generates a 2cm diameter hemisphere, since it would be difficult to print a full sphere without support structures. I argue that printing 2 hemispheres is equivalent to printing a whole sphere since we can just glue them together.

My g-code took a very different approach than the Slic3r generated code. Since we were drawing a sphere, I took advantage of the PrintrBot's implementation of G2 in order to print a series of rings that could create a sphere. The Slicr g-code, on the other hand, produced a set of G1 codes that interpolated a line segment in order to find a circle. Since the firmware already does this by implementing G2 and G3, it seemed wise to take advantage of the firmware's capabilities in order to write less g-code. I used the same preamble and footer as the Slic3r in order to setup the PrintrBot.