Controlling a Soft Materials 3D Printer

Ted Loewenthal

Advisor: Professor G. Krishnan

REU Final Report

Spring 2020

Department of Industrial and Enterprise Systems Engineering

ise.illinois.edu

ILLINOIS

ISE | Industrial & Enterprise Systems Engineering

GRAINGER COLLEGE OF ENGINEERING

TABLE OF CONTENTS

ABSTRACT

The advent of 3D printing represents a tremendous opportunity for the medical simulation community. Precise models of tissue and organs can be modeled in software, and then printed with extreme precision and used to practice difficult and dangerous procedures. However, the vast majority of 3D printers only work with rigid, inflexible materials that are not well-suited for medical applications. To the extent that soft materials 3D printers do exist, they are either extremely expensive or only extrude one material, which limits their usefulness in medical simulations. To that end, the Monolithic Systems Lab has developed its own method of creating a soft materials 3D printer using only components that are available to anyone online. This paper describes the process of reconfiguring the software and circuitry of an inexpensive fused deposition modelling (FDM) 3D printer, with the goal of replacing the default extruder with multiple syringe pumps that can extrude UV-curable elastomer inks. The first step of this process involves wiring to facilitate serial communication between the FDM printer and the new pumps, while accounting for the large voltage difference. Then, the motherboard's firmware is modified and re-uploaded. Lastly, the slicing software on a PC must be updated to enable UV light curing.

1 INTRODUCTION

Most 3D printing requires only a PC and a printer, and follows the specific sequence shown in Figure 1. First, a 3D model is made in CAD software, such as AutoCAD or Solidworks. This model is then uploaded to a “slicer software”, such as Cura or Slic3r, which converts the model into a set of commands in the printer-based programming language G-code. G-code is a set of commands that tells a 3D printer where to move and when to print, as well as modifying settings such as the units or the maximum feedrate [1]. The G-code is then uploaded to the printer, either through a serial connection or an SD card, at which point the printer analyzes these instructions with its on-board firmware and executes them to print the 3D model using its built-in motors.

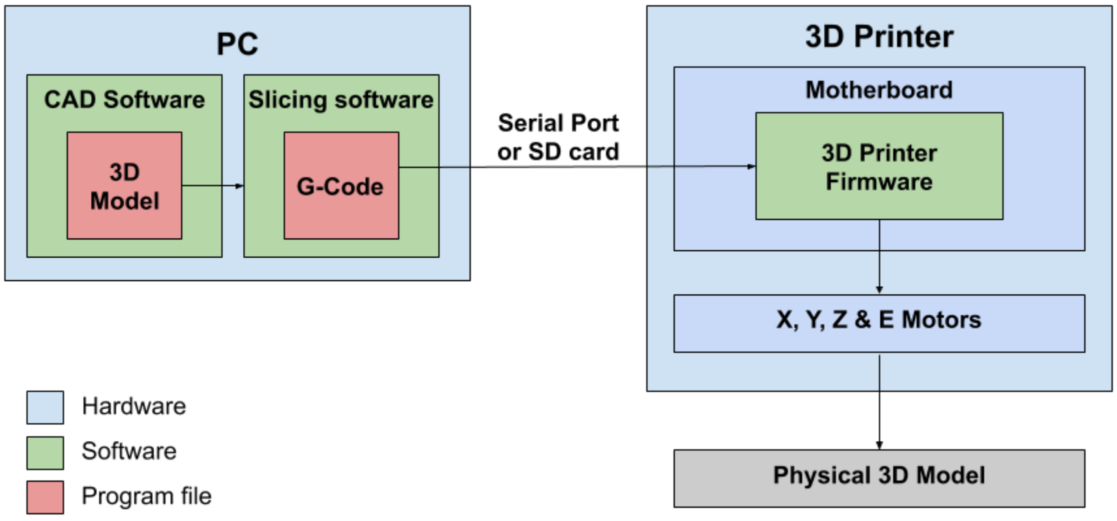

Figure 1: Flowchart showing the relationship between components of a 3D printer.

1.1 Printer Overview

The base 3D printer being modified is the Lulzbot TAZ-6, which was chosen due to its high precision and open-source design, making it an ideal choice for modification. The motherboard used by the TAZ 6 is a RAMBo 1.4, which is essentially an Arduino MEGA expanded to contain five stepper motor drivers on the board. The TAZ-6 comes with X,Y and Z motors connected to an extruder with its own motor (referred to as the E-motor) that pushes out the material based on the build design [2].

The TAZ 6 is a fused deposition modeling (FDM) printer, meaning it extrudes a material at a high temperature that solidifies as it cools. To print soft materials, the TAZ 6 must be transformed into a direct ink writing (DIW) printer that extrudes a fast-curing silicone resin. To accomplish this, two Fusion 6000 syringe pumps have been chosen, which will replace the on-board extruder and extrude two different soft materials that cure under UV light. Fusion 6000 syringe pumps are very durable and extremely precise, which makes them good candidates as extruders for a 3D printer.

This presents a series of issues. Either the printer or the PC must send signals to the new extruders. Either way, this will require physical modifications to the printer. To some extent, the default firmware in the printer must be modified as well. In addition, the slicing software requires modification to enable the UV light curing necessary for this material. These issues, and more, are explored and answered in this report.

2 HARDWARE MODIFICATIONS

There are a few different ways to communicate with the Fusion 6000 extruders. The first method is through the LCD touch screen on the side of the printer. This option is infeasible here because it would require the user to press start and stop hundreds or thousands of times throughout a print. The other methods involve sending serial commands to the pump through either its RS-232 port or its USB port. Through these ports, simple terminal commands such as ‘start’ or ‘stop’ can be sent, and more complicated instructions can be sent via Python or MATLAB scripting [3].

2.1 Sending Serial Commands

The advantage of communicating with the new pumps via simple serial commands is that the commands can be sent from the Arduino chip on the printer. Having the commands come from the PC would require that the printer and the PC sync up to within a fraction of a second, and it would also require wires connecting the PC to two extruders, which would be inconvenient and possibly limited by available ports on the PC. For these reasons, it was decided that the printer would communicate serially with the Arduino on-board the TAZ 6.

2.2 Voltage Conversion

The RAMBo has four rows of pins used to send and receive UART serial commands. However, the serial port on the Fusion 6000 is an RS232, which sends and receives signals at a different voltage level than the RAMBo. To solve this problem, two RS232 to TTL converters are used (BC71075, NKC Electronics). The converters, shown in Figure 2, are built around a MAX232 chip, which uses a capacitive voltage generator to transform the 5V signal from the RAMBo into a 15V signal that the RS232 port can recognize [4]. A more detailed schematic of the converters can be found in Figure A3 in the appendix.

These two converters connect two of the four rows of serial pins in the RAMBo to the two extruders. Hypothetically, if one were to add more extruders for builds that required additional printing materials, a third and fourth extruder could be added by purchasing two additional converters and using the remaining two serial pins.

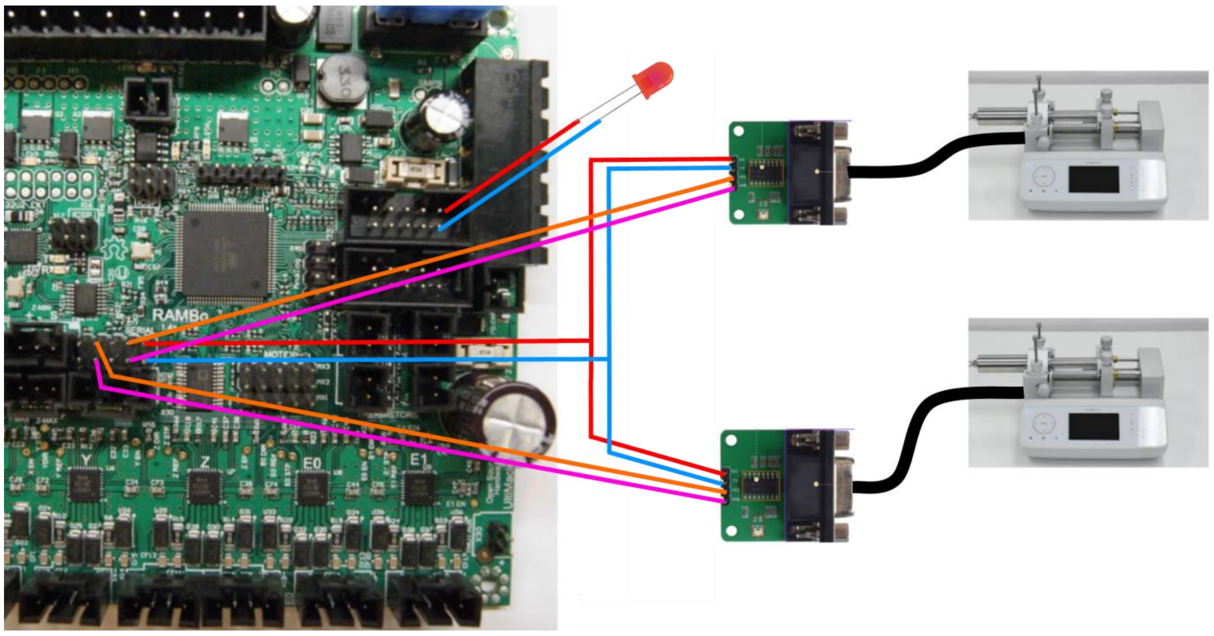

Figure 2: Wiring diagram showing how the Fusion 6000 extruders, the RAMBo serial pins, and the UV LED communicate. Red wires are power, blue wires are ground, orange wires are TX, pink wires are RX, and black wires are DB9 connectors.

2.3 Soldering the UV LED

Modifying the printer physically so it can turn a UV light on and off is rather simple. The RAMBo (Figure A1 in the appendix) has a region of undeclared pins that can be powered on and off via G-code. Leads to the power source for a UV LED array are soldered onto the RAMBo, and the array is positioned on the carriage so that it can shine light upon the entire build at each stage between each printed layer.

2.4 Next Steps

The next step will be to attach the two extruders to a frame that houses the modified printer, with tubes running from the extruders to the TAZ 6's carriage, as shown in Figure 3. This frame will provide both a stable location for the extruders, and a filter for the UV light so as to not cause eye damage.

The flowchart from Figure 1 can be updated to include all of the necessary components for this custom design, as shown in Figure 4.

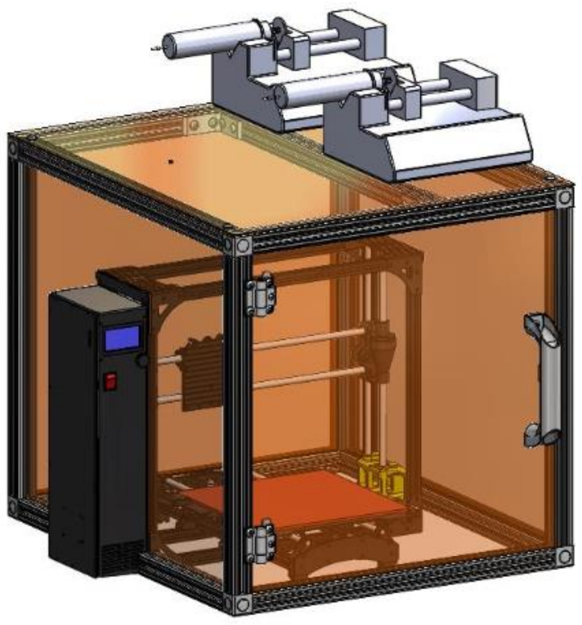

Figure 3: Mockup of the modified 3D printer, with a carriage housing the extruders

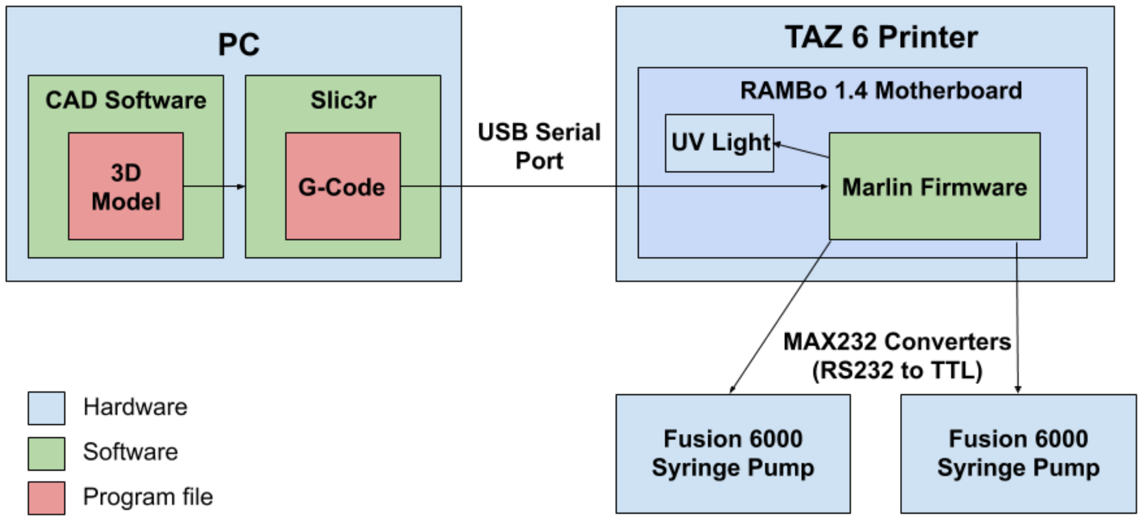

Figure 4: Flowchart for custom soft materials 3D printer, including specific parts.

3 FIRMWARE MODIFICATIONS

The firmware within the RAMBo must be modified in order to replace the movement commands it would send to the default extruder with serial commands sent to the new extruders. The default firmware installed on the RAMBo is Marlin, one of many firmwares designed to interpret G-code into mechanical actions for the motors. Marlin is open-source software written in C++, and can be modified and re-flashed onto the RAMBo via a USB port. For this project, Marlin 1.0.x is used, since it is the easiest version to modify.

3.1 Marlin Code Flow

Marlin's logic (Figure A2 in the appendix) can be broken down into several components, which all feed into its main function, Marlin_main.cpp. The firmware reads blocks of G-code individually, and based on the values seen in the command, the program executes the appropriate operation. The vast majority of G-code operations performed are movement commands, which are interpreted by a series of functions that use movement algorithms to determine the optimal direction, velocity and acceleration of the X, Y, Z and E motors.

3.2 Modified Sections of Code

There are multiple locations where the code must be modified in order to extend its communication to work with the new serial peripherals. Serial communication with two of the four rows of serial pins must first be enabled and then initiated in the main function. The code that enables multiple extruders to function simultaneously must also be modified to work with the new extruders, and send the appropriate serial command to one of them. Since the new extruders will be connected to the same device as the old extruder, the code for the X, Y, and Z motors can all be left untouched.

Marlin also has a function that replaces the default serial communication in Arduino's integrated development environment (IDE) with its own function, Marlinserial.cpp, which it uses to receive G-Code from the PC. To enable the other four serial pins, Marlinserial.cpp must be modified so that it no longer disables the default Hardwareserial file. Having both files enabled simultaneously can be an issue with certain printers, but not the TAZ 6. Once Hardwareserial is re-enabled, the Serial pins (Serial1 and Serial2 in the code) are enabled in the main function with a baud rate of 115200, and the number of extruders is set to 2 in the configuration.h file.

The last step in modifying the firmware to edit is the movement code. The two most convenient locations for manipulating the Marlin movement code are planner.cpp, where decisions about the velocity and acceleration of the motors are made, and stepper.cpp, where those decisions are turned into direct commands to the motors. For the purposes of this project, it is much simpler to modify the motor commands in stepper.cpp, replacing the signals the Arduino was sending to the pins controlling the E-motor with serial signals sent to the two new extruders. These include the pin that controls the direction of the E motor, as well as the pin that controls whether the E motor is on or off.

3.3 Specific Lines of Movement Code Modified

The original E-motor movement commands, and their substitutes, are shown in Table 1. SerialX is either Serial1 or Serial2, depending on which extruder is active. All of these changes are made in a text editor. The modified code is then flashed onto the RAMBo using the Arduino IDE and a USB connector.

Original line of code	**Serial Command Replacement**	**Function in printer**
`WRITE_E_STEP(!INVERT_E_STEP_PIN);`	`SerialX.println("start");`	Starts extrusion
`WRITE_E_STEP(INVERT_E_STEP_PIN);`	`SerialX.println("pause");`	Stops extrusion
`REV_E_DIR();`	`SerialX.println(hexw2 [units] 1 [diameter] [volume] [rate] [delay]);`	Reverse direction of extruder

Table 1: Serial commands sent to the Fusion 6000 extruders

4 MODIFYING SLICING SOFTWARE

The default slicing software that comes recommended for the TAZ 6 is Cura. For this project, Cura is replaced by slicing software Slic3r for the increased amount of G-code customization it provides the user. Within Slic3r, in addition to setting the parameters based on the printer, there is a section for custom G-code that allows the user to add a specific line of G-code at the start of every print, or at the start of every layer [5].

4.1 UV light configuration in Slic3r

For the sake of curing the materials being used, a UV light is connected to the RAMBo and the following line of code is added in Slic3r's printer settings under "custom G-code" to turn the light on and off in between each layer of the build:

```
M41 29 255

G4 S30

M41 29 0
```

M41 is the G-code command to send voltage to a pin, 29 is one of the accessible extension pins on the RAMBo not being used for anything else (highlighted in blue in the appendix in figure A1), and 255 is the maximum voltage that can be sent (5V). G6 will pause the code for the amount of time it takes to cure, which is approximately 30 seconds per layer. Then, the UV light is turned back off and printing can resume.

5 CONCLUSION

The implementation step of this project is currently incomplete, due to interruptions from the COVID-19 pandemic. That being said, the technical issues of facilitating communication between the printer and the new extruders have been solved, via a combination of hardware, on-board software, and slicing software modifications. After quarantine restrictions are lifted, the next steps are to finish wiring both extruders, flash the fully modified firmware onto the printer, and troubleshoot any errors that occur. Upon completion of printer controls, an updated version of this report will include any major changes as well as more detailed implementation instructions.

5.1 Next Steps

Ultimately, these modifications will yield a printer that is capable of printing a wide range of soft materials of different stiffnesses, including a dissolvable support material for more intricate builds. Due to the unique properties of the material used, this printer has applications far beyond that of traditional 3D printers. The full range of these applications is not yet known, due in part to the lack of pre-packaged soft materials 3D printers on the market. The goal of this project is in part to demonstrate that with the proper modifications, one can develop their own soft materials printer, using only publicly available resources. This could be of tremendous benefit to soft robotics researchers and developers of medical simulation devices..

6 REFERENCES

[1] *“G-Code Index*”, Jul. 1, 2016. Accessed on Mar. 2, 2020. [Online]. Available: https://marlinfw.org/meta/gcode/

[2] *Lulzbot TAZ 6 User Manual*, Aleph Objects, Inc., Loveland, CO, 2016. Accessed on: Feb. 17, 2020. [Online]. Available: https://download.lulzbot.com/TAZ/6.0/documentation/manual/TAZ-6-Manual.pdf

[3] *Fusion 4000 & 6000 User Manual*, Chemyx Inc., Stafford, TX, 2016. Accessed on: Feb. 17, 2020. [Online]. Available: https://www.chemyx.com/downloads/Fusion4k6k_Manual.pdf

[4] *MAX232x Dual EIA-232 Drivers/Receivers*, Texas Instruments Inc., Dallas, TX, 2014. Accessed on: Feb. 24, 2020. [Online]. Available: http://www.ti.com/lit/ds/symlink/max232.pdf?&ts=1589237961657

[5] G. Hodgson, *Start, End and Layer Change G-Codes*, Aleph Objects, Inc. Accessed on: Mar. 9, 2020. [Online]. Available: https://manual.slic3r.org/expert-mode/printer-settings

7 APPENDIX

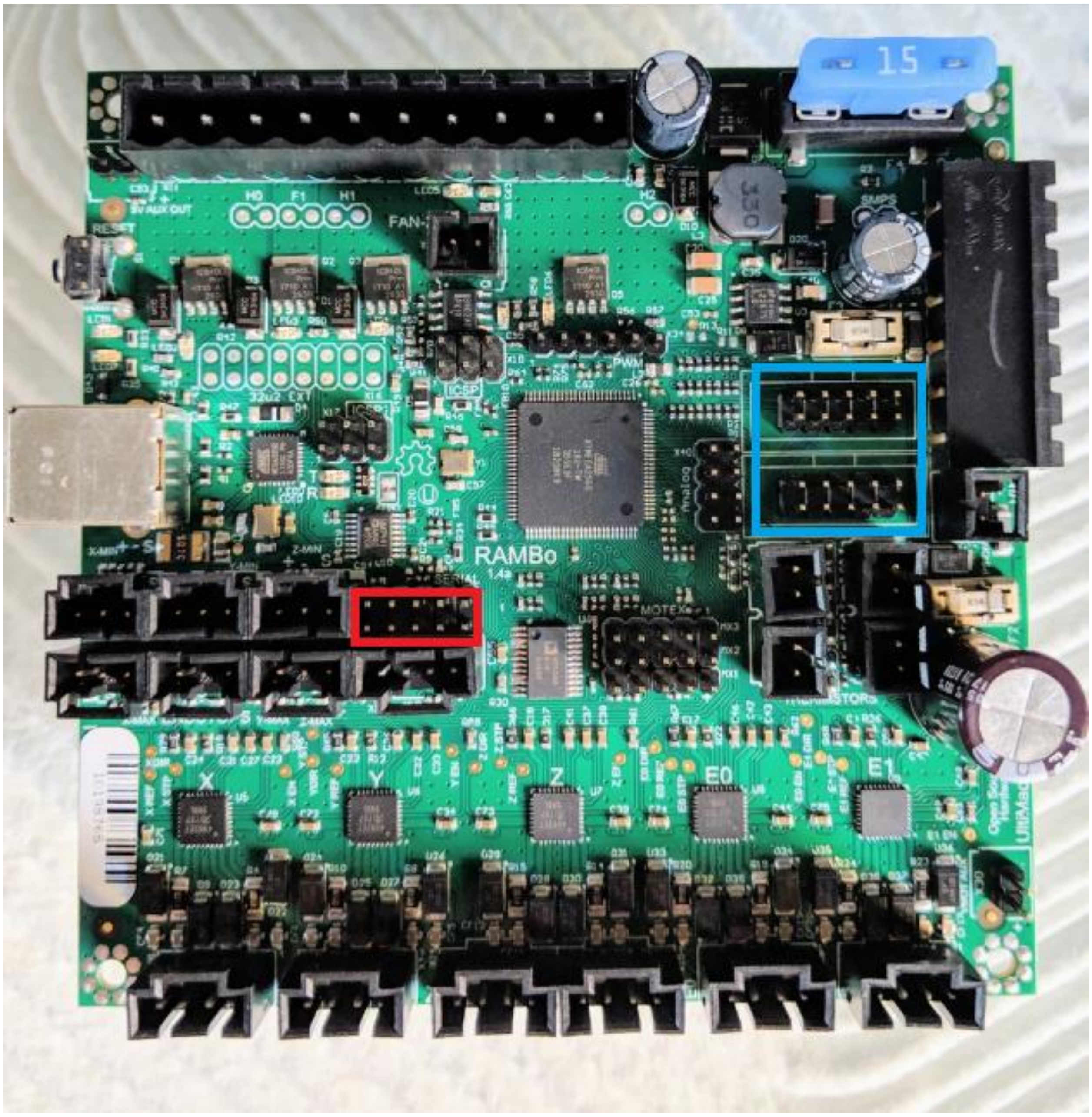

Figure A1: RAMBO 1.4 motherboard. Serial pins highlighted in red, extension pins highlighted in blue.

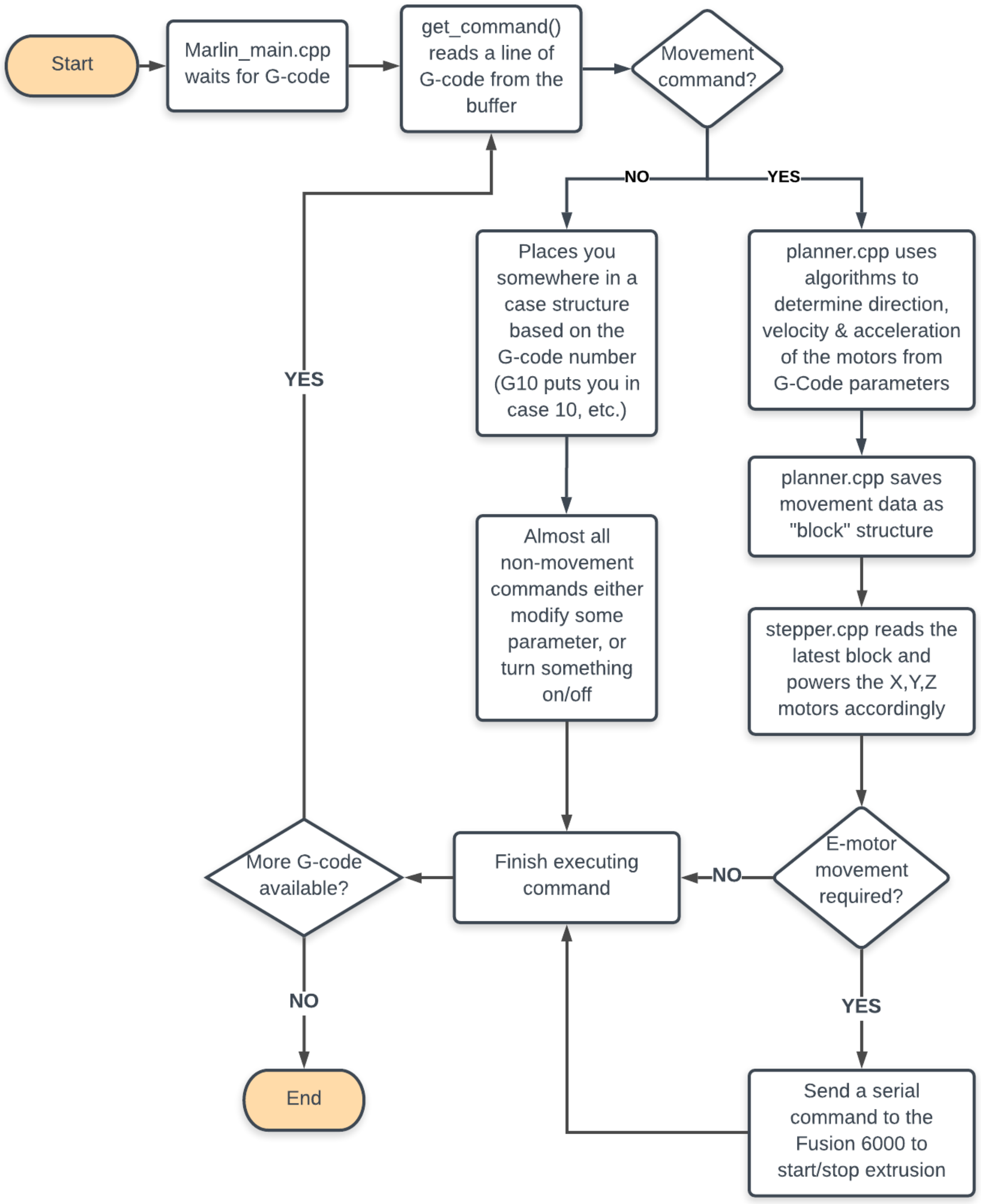
Start
Marlin_main.cpp waits for G-code
get_command() reads a line of G-code from the buffer
Movement command?
NO
YES
Places you somewhere in a case structure based on the G-code number (G10 puts you in case 10, etc.)
planner.cpp uses algorithms to determine direction, velocity & acceleration of the motors from G-Code parameters
planner.cpp saves movement data as "block" structure
Almost all non-movement commands either modify some parameter, or turn something on/off
stepper.cpp reads the latest block and powers the X,Y,Z motors accordingly
YES
More G-code available?
Finish executing command
NO
E-motor movement required?
NO
YES
End
Send a serial command to the Fusion 6000 to start/stop extrusion

Figure A2: Flowchart of Marlin's logic when interpreting G-code.

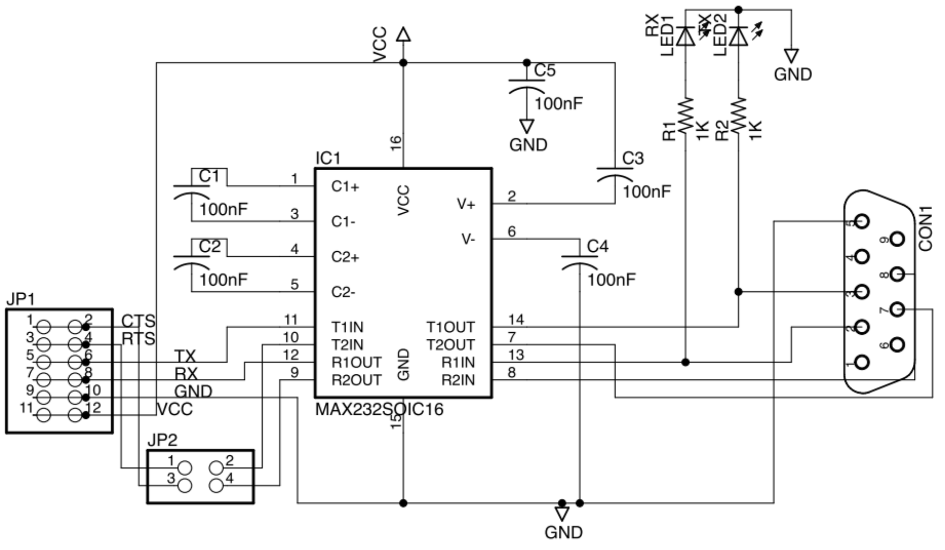

Figure A3: MAX232-based RS232-TTL converter schematic.

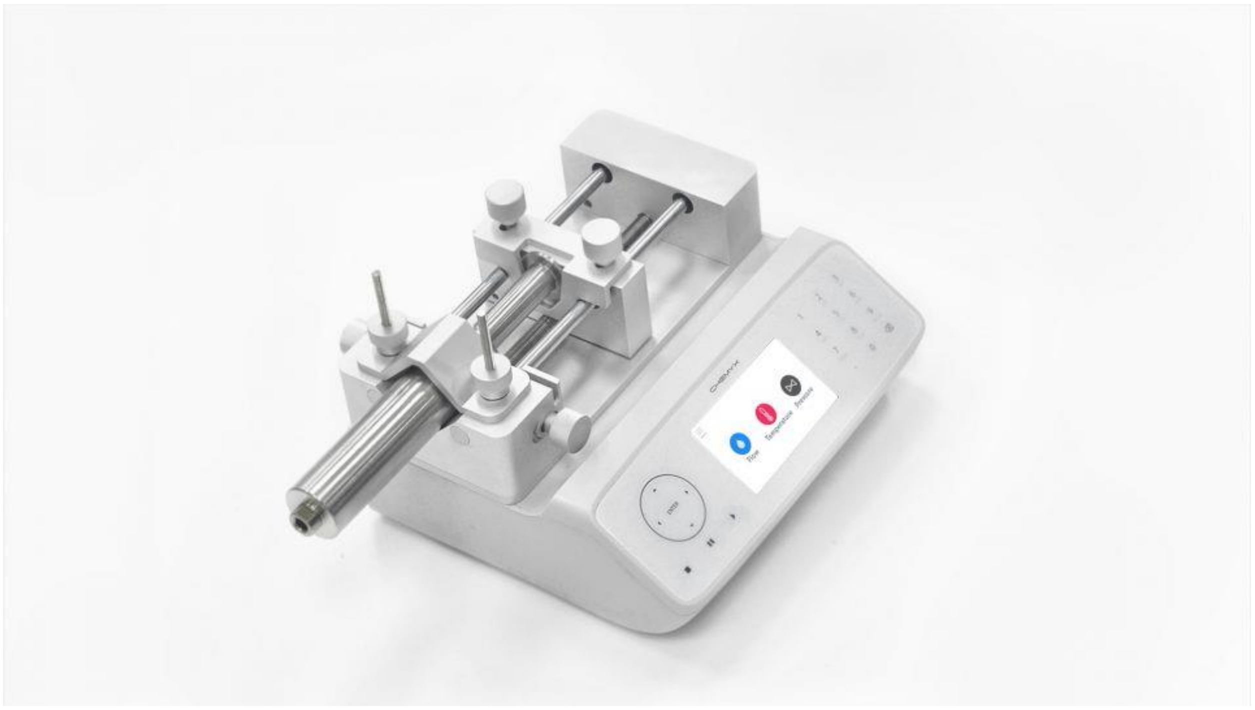

Figure A4: Fusion 6000 extruder.