3D Printer and Application Interface

Cody Hutchison
Steven Liu
Abigail Morar

FINAL REPORT

REVISION – 0.0
4 December 2022

Table of Contents

3D Printer and Application Interface

Cody Hutchison
Steven Liu
Abigail Morar

CONCEPT OF OPERATIONS

REVISION – 0.2
4 December 2022

CONCEPT OF OPERATIONS
FOR
3D Printer and Application Interface

TEAM 21

APPROVED BY:

_____
Cody Hutchison Date

_____
Prof. Lusher Date

_____
Dalton Cyr Date

Change Record

Rev.	Date	Originator	Approvals	Description
0.0	9/14/2022	Dalton Cyr	Cody Hutchison	Draft Release
0.1	9/28/2022	Dalton Cyr	Cody Hutchison	Revision 1
0.2	12/4/2022	Dalton Cyr	Cody Hutchison	Revision 2

Table of Contents

List of Tables

1. Executive Summary

Common 3D printers require local, in-person interaction to start/stop, format settings, or even insert memory devices to upload files. The sponsor for this project, Dalton Cyr, desires a 3D printer capable of printing files with the portability and ease of using a mobile device from any location. With a thoroughly designed phone application and Wi-Fi capability, this 3D printer will have similar features to other standard 3D printers and can also be controlled from any location with internet access. To print, while away from the 3D printer, it must have a server, and to preserve security, the 3D printer will access a web server provided by Firebase which has a reputation for customer security and satisfaction. This project will highlight the pleasing and straightforward user experience of printing 3D designs with the speed and quality of other printers through a mobile phone application.

2. Introduction

This document will review the concepts of operation for a 3D printer and its application interface. While many 3D printers require users to handle their operation in person, this 3D printer will solve this issue with a mobile phone application allowing the user to print from their current location. To achieve this task, the project will be broken down, researched, and executed in several subsystems: power supply, microcontroller unit, dedicated web server, and mobile phone application.

2.1. Background

The combination of a mobile phone application and the dedicated web server will not only provide ease of use, away from the 3D printer but will also act as a simple form of security and solution for transferring large print files. Using the mobile phone application will minimize the need for a user interface on the printer. The only printer control will be the main power switch placing the printer in a standby state. Physical interactions with the 3D printer consist of the initial setup and powering on, adding PLA rolls to the printer, and removing completed prints.

2.2. Overview

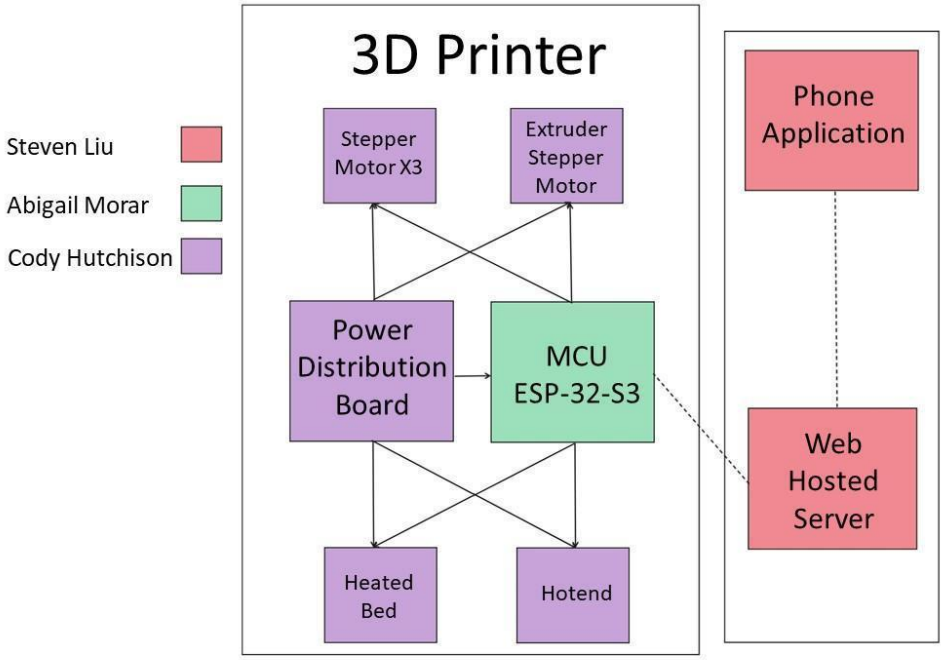


Figure 1. 3D Printer Block Diagram

This 3D printer will be designed to work exclusively through a mobile phone application. The application will receive commands and files and send them to the web-based server setup for the printer. Once uploaded, the server will store the files in the cloud and pass on commands to the motherboard in the 3D printer, if any are received. The motherboard will

then read the codes in order of the most recent upload and begin heating the heated bed and hot head. Once the hot head and heated bed are at the required temperature set by the user, the motherboard will start reading the G-code from the server and initiate printing. The motherboard will also send a code to the server that will be forwarded to the application to start the timer. Another transmitted command would be to cancel/end the print in a situation where there is an issue with the print job, an issue with the design and the person does not want to waste PLA, or the user realized there would not be enough filament to complete the job. Once printing is finished, the printer will send a complete command to the server that will be forwarded to the application to notify the individual that the print is done, and the timer will be stopped. Power will be distributed to the motherboard, the stepper motors, the hot head, the heated bed, and the extruder for operation and communication between subsystems.

2.3. Referenced Documents and Standards

- IEEE Wi-Fi Std 802.11-2020
- IEEE Standard for Safety Levels with Respect to Human Exposure to Electric, Magnetic, and Electromagnetic Fields, 0 Hz to 300 GHz, ANSI C95.1-2019
- Android App Development, “Core App Quality” https://developer.android.com/docs/quality-guidelines/core-app-quality

3. Operating Concept

3.1. Scope

The goal is to develop a 3D printer that is controlled by a phone application that stores the G-code files on an internet server and allows the printing of the G-code files from anywhere that has internet access.

3.2. Operational Description and Constraints

This 3D printer system requires internet access from both the 3D printer and mobile device. Before using the 3D printer, the microcontroller unit will require an initial connection to the web-hosted server and local network at its location with its main power switch on standby. The user will send G-code files and printer commands from the mobile phone application to the server. On the printer side, the microcontroller will send pull requests in intervals receiving packets of the files/commands to control the 3D printer while also sending push requests of collected data to the server. This data is displayed through the mobile application allowing the user to monitor the 3D printer's progress. The 3D printer design must meet the following criteria:

- Operate within 10-15% speed of the initial hardware and software used for the Ender 3 3D printer
- Print area of 220mm x 220mm x 250mm
- Print height of 0.2 mm per layer
- Use 1.75mm PLA for print material
- The heated bed must reach a maximum temperature of 80℃
- The hot head must reach a maximum temperature of 220 ℃

The printer will have an input of 120V 60Hz AC power that will be converted to 24V 15A. The system will use an ESP32 microcontroller to communicate with the host server and provide commands to the motherboard. The interface for the 3D printer will be through an android mobile application with the host server designed through Firebase cloud services.

3.3. System Description

There are three main subsystems for the 3D printer and mobile phone application. The first subsystem is the interface consisting of the mobile phone application and the web server. The mobile phone application will take G-code files and upload them to a web server and allow the initiation and shutdown of the 3D printer. The application will also handle initializing specific printing specifications desired by the user such as heated bed/hot head temperatures and stepper motor adjustments. Other features of the application are keeping track of print progress, the amount of material used, and housing a library of prints sent from the phone. The web host server will act as an intermediary for file transfer and controls between the phone application and the 3D printer while also housing previous 3D print files.

The second subsystem is the microcontroller for the 3D printer. While in standby mode, the microcontroller will send requests in specified intervals to take in commands from the cloud server and send them to the motherboard. Such commands may include the starting/stopping of printing, the control over fan speed, the location of the hot head, the

amount of extruded material, the retraction of filament, and the temperature of the heated components. The motherboard will process and distribute these commands whether in standby or active printing. The movement of the stepper motors will be communicated to the stepper motor control board and adjusting the temperature of the heated bed and hot head will be communicated to their boards, respectively. The microcontroller will also collect information to be sent back to the phone application allowing the user to monitor the printing progress and selected print settings.

The third subsystem will be the motor controller boards, temperature controlling boards, and a power distribution board. The motor controller boards will take commands from the microcontroller and send power to the stepper motors by use of dual H-bridges. The temperature control board will also send power to the heated bed and hot head by the use of MOSFETs and signals from the microcontroller. The power distribution board will take the 24V 15A, split it up, and send 5V 1.5A to all 4 stepper motors; 5V 0.5A to the microcontroller, 24V 9.167A to the heated bed, and 24V 1.67A to the hot head.

3.4. Modes of Operations

There is a single planned mode of operation for the 3D printer electronic system. That operation is to control the 3D printer through a mobile phone application that requires only a main power switch on the printer to stop all external power.

3.5. Users

This 3D printing system is aimed toward hobbyist 3D printers. The setup requires basic knowledge to connect the printer's built-in microcontroller to the web server and local network by following the instructions in the manual. After the server's setup is complete, anyone with proper G-code files and an internet connection can use the printer.

3.6. Support

The manual will provide assembly instructions for the 3D printer as well as instructions to set up the connection to the web server and local network. The manual will also display the mobile application name and link to download it. The app will provide a walkthrough of all its features along with how to send print files, adjust heating components or extruder positioning, and start or stop a print design.

4. Scenario(s)

4.1. Parts Rush

- Parts rush required due to competition schedule and broken parts so prints may be initiated while traveling from a competition.

4.2. 3D Printing Business

Need to start a print order while out of the office to meet customer deadlines.

4.3. Work Prototyping

- Modifications are required on a prototype 3D print that cannot be finished at work. Allows employees to finish adjustments and initiate printing from home.

4.4. Hobby Printing

- Allows hobbyists to print from any location as long as there is internet connection.

5. Analysis

5.1. Summary of Proposed Improvements

Improvements include user-friendly setup, ease of feature adjustments, print progress notifications, the ability to start and stop jobs remotely, virtually infinite file storage, faster auto-home command, and convenient communication between devices.

5.2. Disadvantages and Limitations

Feeding issues could be a major disadvantage to this system. This would be caused by a clog in the extruder or the 3D printer running out of filament. The limitation of having only a phone application to communicate with the 3D printer would prove to be a problem if the local network is ever down since the device can only print files when connected over the internet.

5.3. Alternatives

One alternative is that the printer could be controlled through Bluetooth instead of a network connection to the server. A drawback to that would be the need for users to stay within range of the printer to control and operate it.

5.4. Impact

Possible impacts:

The capability to 3D print from anywhere, allows people to be more social and not be confined to their homes to print an item.

With the freedom of not having to constantly monitor the printer, the individual can now volunteer to clean up around their neighborhood, town, or city.

By making the 3D printer user-friendly, it allows people who are not technically inclined to a platform to perform 3D printing.

An ethical issue would be leaving the printer to run unattended with the 80℃ heated bed and a 200℃ hot head. A fire could occur if a flammable item fell onto the hot head or heated bed. Without supervision, these heated components could cause dangerous outcomes such as a major injury to another person or pet.

3D Printer and Application Interface

Cody Hutchison
Steven Liu
Abigail Morar

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – 0.1
4 December 2022

FUNCTIONAL SYSTEM REQUIREMENTS
FOR
3D Printer and Application

TEAM 21

_____
Author Date

APPROVED BY:

_____
Cody Hutchison Date

_____
John Lusher, P.E. Date

_____
Dalton Cyr Date

Change Record

Rev.	Date	Originator	Approvals	Description
0.0	10/3/2022	Dalton Cyr	Cody Hutchison	Draft Release
0.1	12/4/2022	Dalton Cyr	Cody Hutchison	Revision 1

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Purpose and Scope

The 3D Printer will provide an efficient solution to users who desire to 3D print remotely. This project shall incorporate the use of a mobile application, Firebase cloud computing service, and onboard internet capabilities along with common 3D printing hardware to achieve this goal. The 3D printer shall retain the same options as the printer's original hardware and software offered. The project will be divided into several subsystems listed below in Table 1.

Figure 1. Ender 3 3D printer

1.2. Responsibility and Change Authority

The team leader, Cody Hutchison, will ensure all project requirements will be met and that the team will keep to the planned schedule. All subsystems and respective responsibilities are listed below in Table 1. Any and all changes must be approved by the team leader and Sponsor, Dalton Cyr. Cody Hutchison is responsible for the stepper motor controller, the heating elements of the hot head and hot bed, and the correct power distribution to all the required hardware components. Steven Liu is in charge of the development of the 3D printer application and making sure it connects to the web-hosted server. He is also responsible for the development of the server which will communicate with the MCU. Abigail Morar is responsible for the microcontroller and ensuring that it is sending and receiving signals appropriately, as well as communicating between each component and subsystem effectively.

Subsystem	Responsibility
Power Distribution	Cody Hutchison
Onboard Hardware	Cody Hutchison
Onboard Software	Abigail Morar
AWS Server	Steven Liu
Mobile Application	Steven Liu

Table 1. Subsystems and Responsibilities

2. Applicable and Reference Documents

2.1. Applicable Documents

Document Number	Revision/Release Date	Document Title
Ender 3 Printer Manual		Ender 3 Printer Manual
ESP32-S3 Series	V1.3	ESP32-S3 Series Datasheet
802.11-2020	08 October 2021	IEEE Standard of for Information Technology

Table 2. Applicable Documents

2.2. Reference Documents

Document Number	Revision/Release Date	Document Title
N/A	N/A	Android App Development

Table 3. Reference Documents

2.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings, or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

3.1. System Definition

The project is to design a control system and application that allows the printer to be operated by an application from anywhere there is internet access.

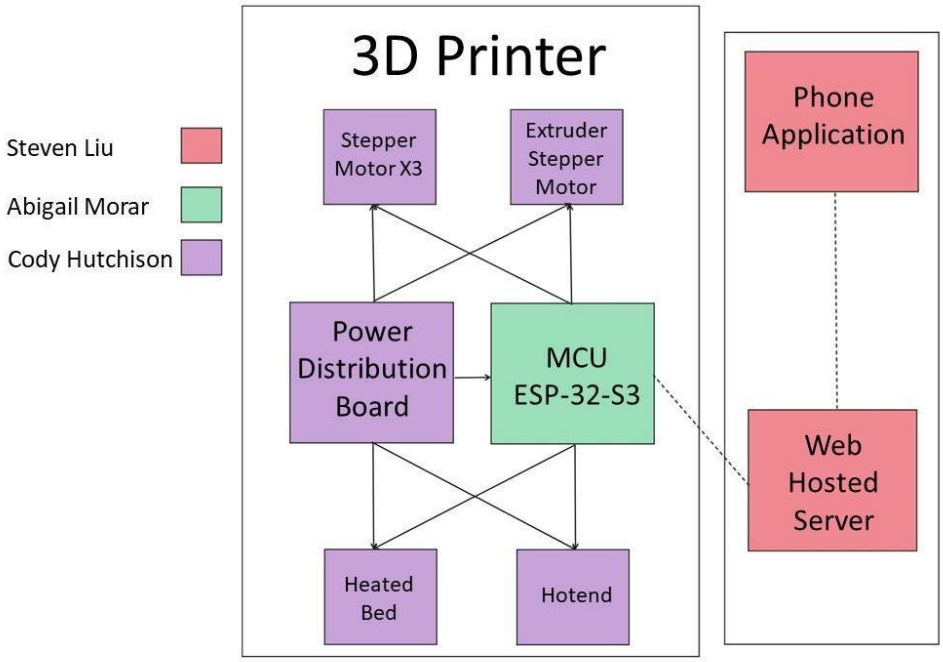


Figure 2. Block Diagram of System

The three main subsystems are the MCU programming, the android application and web server, and finally the control and power distribution board. The reason for changing the 3D printer design is to convert it from being manually operated through the application only. The original hardware was all combined into two boards which is the reason for building all the different boards.

3.2. Characteristics

3.2.1. Functional / Performance Requirements

3.2.1.1. Print Time

Constraining the print time within 10-15% of the time it takes to print with the original hardware and software.

Rationale: This is a requirement specified by the customer to meet the capabilities set by the old software.

3.2.1.2. Axis Speed

Increase the Z-axis speed so that it is closer to the speed of the X and Y-axis.

Rationale: By increasing the Z-axis speed allows the movement of the hot head and home return.

3.2.1.3. Temperature Control

The heated bed needs to be controllable from 0-80℃ and the hot head needs to be controllable from 0-220℃.

Rationale: This is a requirement specified by the customer to have the heated components be heated enough for PLA filament to be properly used. These temperature ranges allow adjustability for different brands of PLA.

3.2.1.4. Printing Layer Height

Each layer will have a height of 0.2mm.

Rationale: This is a requirement specified by the customer since this is the common printing height of the filament.

3.2.2. Physical Characteristics

3.2.2.1. Print Area

The printing area will be 220mm x 220mm x 250mm.

Rationale: This is a requirement specified by the customer due to the printing area of the original 3D printer base/frame being used.

3.2.2.2. Printing Material

The printing material needs to be 1.75mm PLA.

Rationale: This is a requirement specified by the customer since this printing material is common and affordable.

3.2.3. Software Characteristics

3.2.3.1. Mobile Application

The mobile application shall be developed for Android devices.

Rationale: The application is done for Android devices due to an accessible developmental environment and available tools.

3.2.3.2. Mobile Application Software

The mobile application software shall be developed using the Kotlin programming language.

Rationale: The Kotlin language is chosen due to its compatibility with android application development.

3.2.3.3. Web-Hosted Server

The intermediary server shall be developed using Firebase cloud services.

Rationale: Firebase is chosen for its reputable performance, quality and security. Firebase also has a wide variety of integrable services that meet the needs of the customer.

3.2.4. Electrical Characteristics

3.2.4.1. System Power

3.2.4.1.1 Power Consumption

The total power consumption shall not exceed 360 W.

Rationale: The maximum amount of power consumption was below 360 Watts, and the maximum output of our power supply is 360 Watts.

3.2.4.2. Outputs

3.2.4.2.1 Data Output

The 3D Printing System shall include an interface to view the printing progress.

Rationale: The 3D printer sends progress data back to the user through the server and mobile application.

3.2.5. Failure Propagation

3.2.5.1. Failure Detection

The 3D Printing System shall have a software subsystem in the server to check if the G-code files to be printed will fit within the 3D printing size constraints and if the heated elements are within proper printing temperatures.

Rationale: This is a requirement to avoid oversized prints due to large print designs and poor adhesion of print layers due to inadequate extrusion and hot bed temperatures.

3.2.5.2. Failure Warning

The 3D Printing System will record how much filament has ever been used to notify the user of potential print failure when filament is estimated to run out.

Rationale: Users will be able to account for and resolve problems when they are planning to print remotely.

4. Support Requirements

The 3D printer and application will require an internet connection and access to 120VAC 60Hz power. The web-hosted server will also require an initial setup through the Firebase web-hosted server website. The 3D printer will require some assembly. If taken to a new location where the local network is changed, a new connection setup with the web-hosted server will have to take place. The phone application will need to be downloaded onto an android device with Wi-Fi/internet connection capability.

- # Appendix A: Acronyms and Abbreviations

3D	3 Dimensional
A	Amperage
C	Celsius
FRD	Firebase Realtime Database
FS	Firebase Storage
G-code	Geometric Code
GHz	Gigahertz
Hz	Hertz
MHz	Megahertz
ICD	Interface Control Document
IEEE	Institute of Electrical and Electronic Engineers
MCU	Microcontroller Unit
mm	Millimeter
oz	Ounce
PCB	Printed Circuit Board
PLA	Polylactic Acid
URL	Uniform Resource Locator
V	Voltage
VAC	Volts Alternating Current
W	Watts

- # Appendix B: Definition of Terms

- # Appendix C: Interface Control Documents

3D Printer and Application Interface

Cody Hutchison
Steven Liu
Abigail Morar

INTERFACE CONTROL DOCUMENT

REVISION – 0.1
4 December 2022

INTERFACE CONTROL DOCUMENT
FOR
3D Printer and Application

TEAM 21

_____
Author Date

APPROVED BY:

_____
Cody Hutchison Date

_____
John Lusher II, P.E. Date

_____
Dalton Cyr Date

Change Record

Rev.	Date	Originator	Approvals	Description
0.0	10/3/2022	Dalton Cyr	Cody Hutchison	Draft Release
0.1	12/4/2022	Dalton Cyr	Cody Hutchison	Revision 1

Table of Contents

List of Tables

List of Figures

1. Overview

The interface control document details how the microcontroller is interfacing with all the other control boards, the web-hosted server, and the application.

2. References and Definitions

2.1. References

Refer to section 2.2 of the Functional System Requirements document.

2.2. Definitions

Refer to Appendix A of the Functional System Requirements document.

3. Physical Interface

3.1. Weight

Component	Weight (oz)	Number of Items
ESP32-S3	0.4	1
Stepper Motor Control Board	1.3	1
Main Control Board	4	1

Table 1. PCB Weight

3.2. Dimensions

Component	Length (mm)	Width (mm)
ESP32-S3	62.74	25.4
Stepper Motor Control Board	148.082	57.277
Main Control Board	152.4	101.6

Table 2. PCB Dimension

4. Electrical Interface

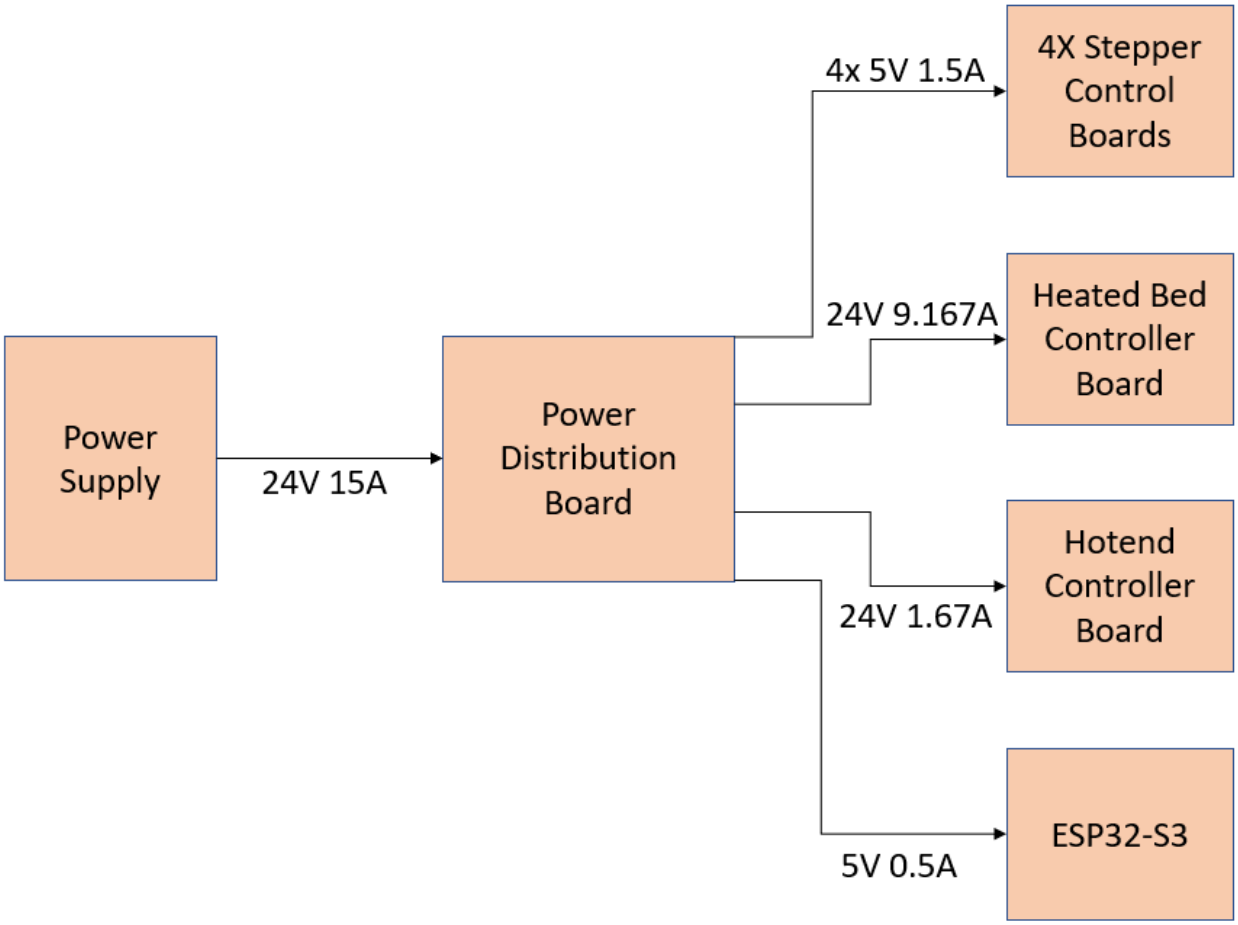


Figure 1. Electrical Interface Diagram

4.1. Primary Input Power

4.1.1. Power Supply

The power supply coming in will be 24V 15A and is the original power supply that the 3D printer had.

4.1.2. Power Distribution

The power distribution board will take the 24V 15A and distribute the power required to each board.

4.2. Voltage and Current Levels

Components	Voltage (V)	Max Current (A)	Max Power (W)
ESP32-S3	5	0.5	2.5
Stepper Motor Control Board	5	1.5	7.5
Heated Bed Control Board	24	9.167	220
Hot Head Control Board	24	1.67	40
Power Distribution Board	24	15	360

Table 3. Component power usage

4.3. Signal Interfaces

4.3.1. ESP32-S3 Signal Interface

The ESP32-S3 will be controlled by a built-in 802.11 b/g/n 2.4 GHz Wi-Fi + Bluetooth module. It will be connected to the server and motherboard.

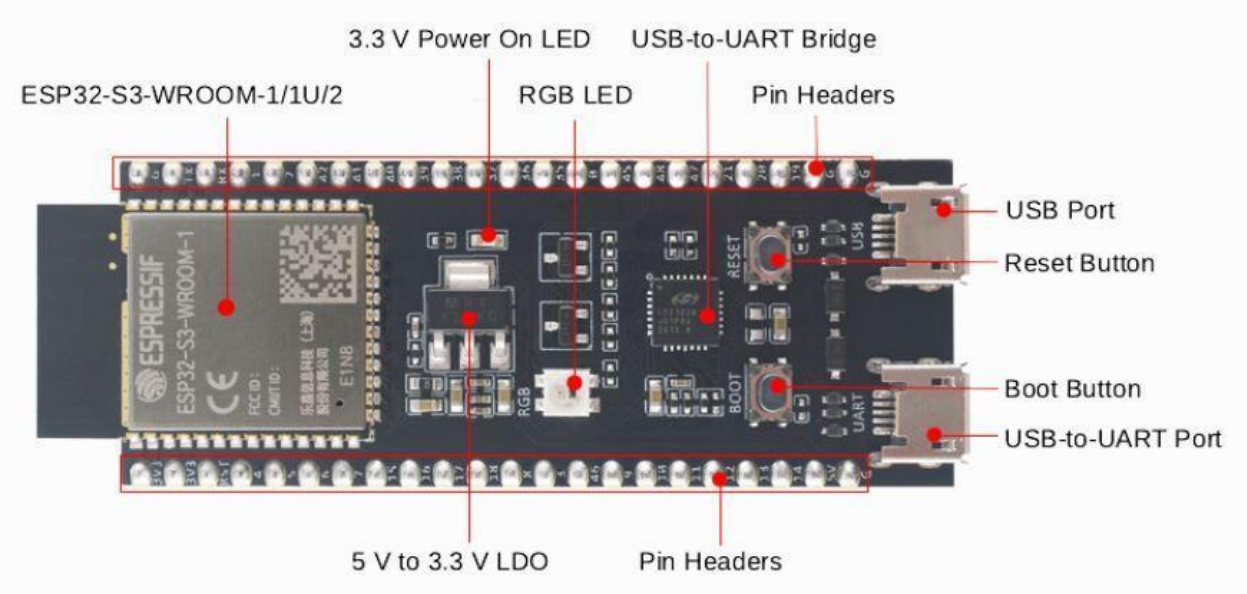


Figure 2. ESP32-S3 Layout

5. Communications / Device Interface Protocols

5.1. Wireless Communications (Wi-Fi)

The ESP32-S3 has a Wi-Fi module built into the board that follows the IEEE 802.11 b/g/n standard protocol. This will allow the microcontroller to be controlled by the application through the Wi-Fi and web-hosted server.

3D Printer and Application Interface

Cody Hutchison
Steven Liu
Abigail Morar

SCHEDULE AND VALIDATION PLAN

REVISION – 0.1
4 December 2022

SCHEDULE AND VALIDATION PLAN
FOR
3D Printer and Application

TEAM 21

_____
Author Date

APPROVED BY:

_____
Cody Hutchison Date

_____
John Lusher, P.E. Date

_____
Dalton Cyr Date

Change Record

Rev.	Date	Originator	Approvals	Description
0.0	10/3/2022	Dalton Cyr	Cody Hutchison	Draft Release
0.1	12/4/2022	Dalton Cyr	Cody Hutchison	Revision 1

Table of Contents

List of Tables

1. Execution Plan

3D Printer and Application

Completed | In-Progress | Not Started | Over-Schedule

Activity	Plan Start	Plan Duration	Actual Start	Actual Duration	Percent Complete
Status Update 1					
Finalize MCU Board Choice	12/Sep	1	12/Sep	1	100%
Finalize Application Choice	12/Sep	1	12/Sep	1	100%
Design Stepper Motor Controller	12/Sep	3	12/Sep	6	100%
Design Application	19/Sep	4	26/Sep	11	100%
Status Update 2					
Finalize Server Choice	3/Oct	2	26/Sep	2	100%
Design Heated Bed and Hotend Controller	3/Oct	1	3/Oct	5	100%
Design Power Distribution	10/Oct	3	17/Oct	2	100%
Coding Microcontroller	10/Oct	7	13/Oct	6	100%
Design Web-Hosted Server	10/Oct	4	10/Oct	4	100%
Status Update 3					
Design PCB	3/Oct	2	5/Oct	4	100%
Validate Mobile Application	24/Oct	1	24/Oct	6	100%
Validate Application and Server Interconnection	30/Oct	1	30/Oct	5	100%
Solder PCB	14/Nov	1	21/Nov	1	100%
Status Update 4					
Test/Validate Power Distribution Board	30/Oct	1	7/Nov	1	100%
Test/Validate MCU	17/Oct	2	20/Oct	1	100%
Test/Validate Stepper Motor Controller	10/Oct	1	24/Oct	2	30%
Test/Validate Heated Bed and Hotend Controller	24/Oct	1	30/Oct	1	100%

Date: 29/Aug | 5/Sep | 12/Sep | 19/Sep | 26/Sep | 3/Oct | 10/Oct | 17/Oct | 24/Oct | 30/Oct | 7/Nov | 14/Nov | 21/Nov | 28/Nov

Table 1. Gantt Chart

2. Validation Plan

FSR ref.	Validation	Success Criteria	Test Bench	Status	Responsible
3.2.1.2.	Stepper Motor Circuit	Output of 5V 0.84A on three specific outputs, 5V 1A on another output with specific controller inputs	Use Function Generator, DC power generator, controller board, and multimeter to validate output requirements.	Tested Not Validated	Cody Hutchison
3.2.1.3.	Heated Bed Circuit	Output of 24V 9.167A when controller input is made	Use Function Generator, DC power generator, controller board, and multimeter to validate output requirements.	Tested	Cody Hutchison
3.2.1.3.	Hotend Circuit	Output of 24V 1.67A when controller input is made	Use Function Generator, DC power generator, controller board, and multimeter to validate output requirements.	Tested	Cody Hutchison
3.2.1.3.	Fan 1 and 2 Circuit	Output of 24V 100mA when controller input is made	Use Function Generator, DC power generator, controller board, and multimeter to validate output requirements.	Tested	Cody Hutchison
3.2.4.1.	Power Distribution Circuit	Ensure DC-DC convertor takes 24V input and outputs 5V	Use multimeter to validate all output possiblilites and connections.	Tested	Cody Hutchison
3.2.3.1.	Full Printer Control Features	User choosen printer formatting, file selection, and print initiation	Input printer formatting, select print file, select print initiation and check server for verification	Tested	Steven Liu
3.2.3.2.	Application File Chooser	Application succussfully selects device files	Perform file upload to server and check firebase storage for verification	Tested	Steven Liu
3.2.3.3.	Interactive Library View	Library displays print files that users can select	Attempt print initiation and check server and home tab for verfication	Tested	Steven Liu
3.2.3.4.	Server Storage	Stores print files	Upload print file and retrieve download url	Tested	Steven Liu
3.2.3.5.	Server Realtime Database	Saves print data and accessible print data	attempt printer formatting change and file upload, verify data in server and library tab	Tested	Steven Liu
3.2.3.6.	Application and Server Connection	Successful communication between application and server	Attempt all application printer features and verify communication to server	Tested	Steven Liu
3.2.4.2.	Communication of ESP32	Check incoming transmission from server	Connect MCU to network and send pings to server.	Tested	Abigail Morar
3.2.1.4.	Extruder	Ensure printer is extruding and retracting the proper amount of filament	Feed filament through opening and send code to force filament through nozzle by specified amounts.	Untested	Abigail Morar
3.2.1.	Microcontroller	Can communicate correctly with motors and heated components	Sending inputs through every channel to show everything is connected and operating correctly.	Untested	Abigail Morar
3.2.1.1.	Stepper Motors	Function smoothly and rotate accordingly by required distance	Various input cases will be used to track movement of the motors. This will be tested to ensure correct communication between board and device.	Untested	All
3.2.1.3.	Heated Bed Temperature	Can reach 220°C	Use infrared thermometer to check surface temperature.	Untested	All
3.2.1.3.	Hotend Temperature	Can reach 80°C	Use infrared thermometer to check surface temperature.	Untested	All
3.2.2.1.	Extruder Location	Can reach the full printing area of 220x220x250 (mm)	Set extrusion nozzle head to every possible coordinate.	Untested	All

Table 2. Validation Table

3D Printer and Application Interface

Cody Hutchison
Steven Liu
Abigail Morar

SUBSYSTEM REPORTS

REVISION – 0.0
4 December 2022

SUBSYSTEM REPORTS
FOR
3D Printer and Application Interface

TEAM 21

APPROVED BY:

_____
Cody Hutchison Date

_____
Prof. Lusher Date

_____
Dalton Cyr Date

Change Record

Rev.	Date	Originator	Approvals	Description
0.0	12/4/2022	Dalton Cyr	Cody Hutchison	Draft Release

Table of Contents

List of Tables

List of Figures

1. Introduction

The 3D Printer will provide an efficient solution to users who desire to 3D print remotely. This project shall incorporate the use of a mobile application, Firebase cloud computing service, and onboard internet capabilities along with common 3D printing hardware to achieve this goal. The 3D printer shall retain the same options as the printer's original hardware and software offered. The subsystems are broken down into the Printed Control Board, Microcontroller Programming, and the Phone Application and Web-based Server. With all three subsystems being meticulously designed and rigorously tested to allow for smooth integration of the three systems together in accordance and to the specifications as laid out in the CONOPS, FSR, and ICD.

2. Printed Control Board Report

2.1. Printed Control Board Introduction

The two printed control boards are what will take the digital signal from the Microcontroller and allow the 3D printer to work. The power for all the 3D printer components, the MCU, and the PCB's will be from a 24V 360W power supply that came with the printer. The PCBs were evaluated and validated to ensure that all systems received the power they required and operated as designed and expected to.

2.2. Printed Control Board Details

2.2.1. Main Control PCB

There were multiple goals with the Main Control board. The first was to receive the power from the power supply and distribute and convert it to 5V and then send it to 3D printer Components. Stepper motor controllers, and MCU. The second was to mount the MCU on the board to allow signals to be sent out to the controllers. The third and final was to house the heated bed, hot head, fan 1 and 2, thermistors, and end switches all on the signal board.

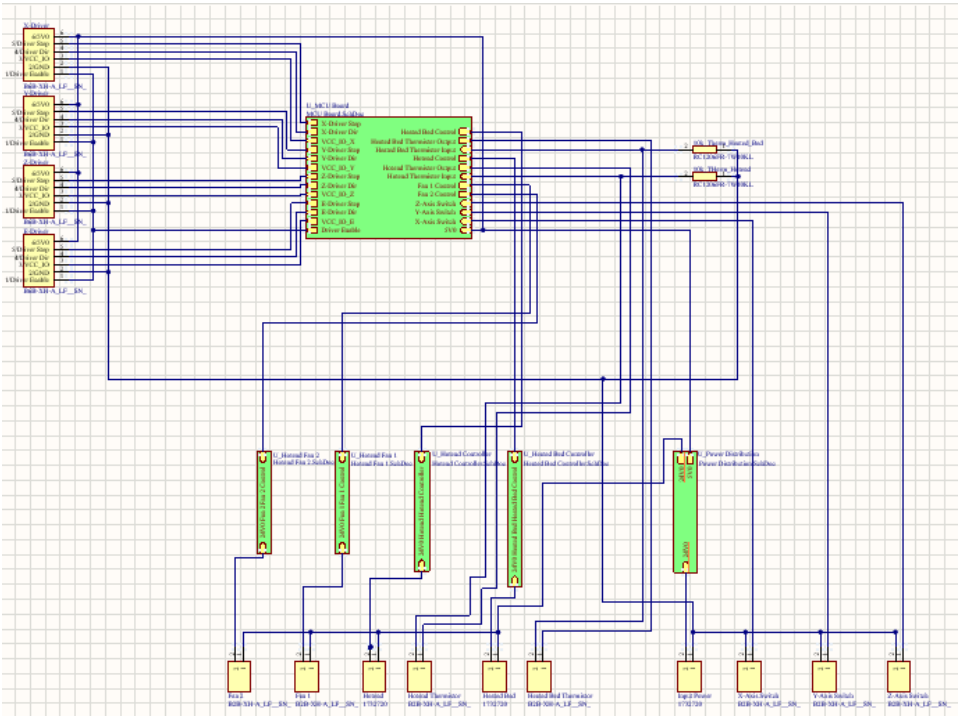


Figure 1. Main Control PCB Schematic

2.2.2. Stepper Motor Controller PCB

The stepper motor controller PCB is designed to take inputs from the MCU that travel through connectors to the PCB and output them into stepper motor movements. The PCB holds four separate controllers that will control the X-axis, Y-axis, Z-axis, and extruder stepper motors. When the printer is in standby mode or is not printing the stepper motor controllers receive a signal from the MCU to the driver enable pin of the controller to put the chip into standby mode. The system is evaluated to ensure it takes in a square waveform from a waveform generator, 3.3V from a power supply, and 5V from the Main Control PCB and causes the stepper motor to move.

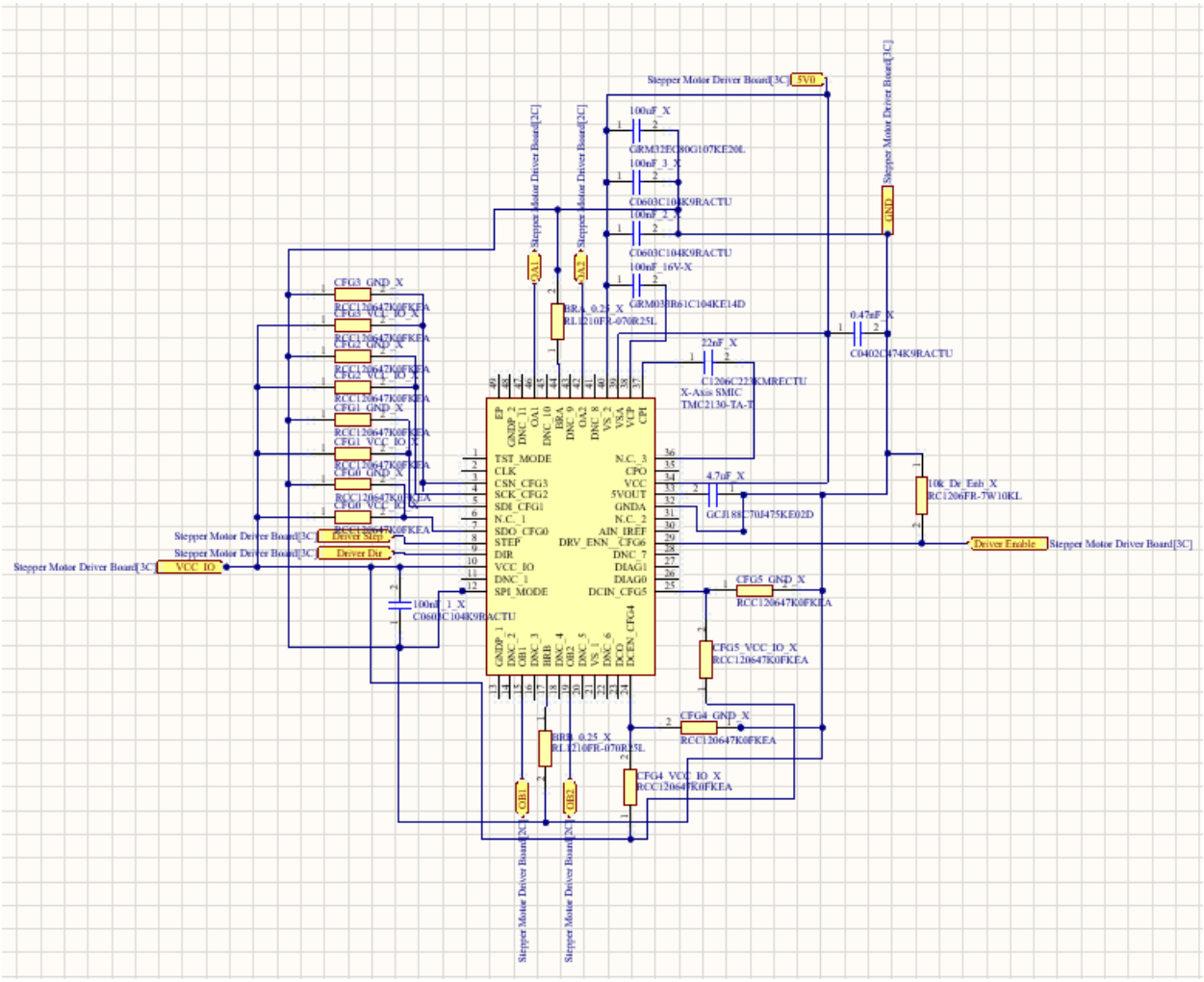


Figure 2. Stepper Motor Controller PCB Schematic

2.3. Printed Control Board Validation

2.3.1. Main Control PCB

To start the validation and test of the main control PCB was to attach the power supply and make sure the appropriate power was going where it was required. The power supply was

also checked with an oscilloscope at the PCB input to ensure there wasn't too much noise. Once that was accomplished then there was the need to test each controller to make sure that they were working properly and wouldn't burn up due to the current load.

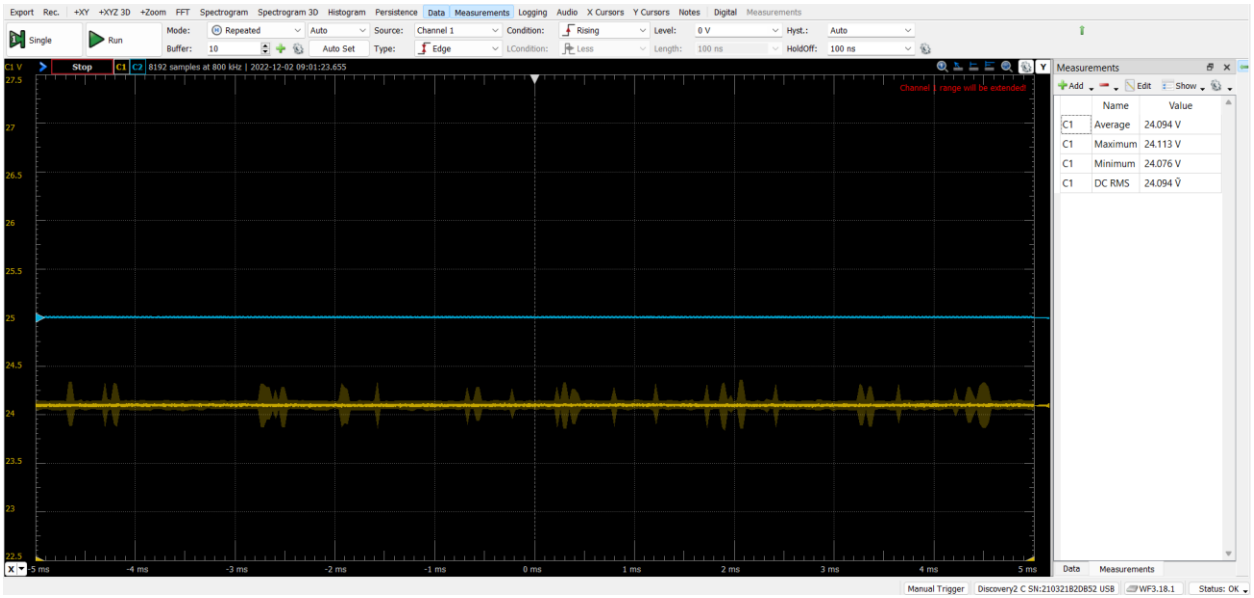


Figure 3. Power Supply input into Main Control PCB

The first circuit test was done on the heated bed controller. 3.3V were sent to the gate of the N-type mosfet which allowed a current to flow through the heated bed and into the ground allowing the heated bed to heat up. The first data point I collected was the voltage by using my multimeter and touring the ground to the ground and the positive to the positive. I then tested the current by disconnecting the heated bed ground wire and connecting my positive lead to the wire and the ground lead to the ground post of the phoenix connector being used for testing. The same process was used to test the hot head and both fans and all the data points are recorded in table 1.

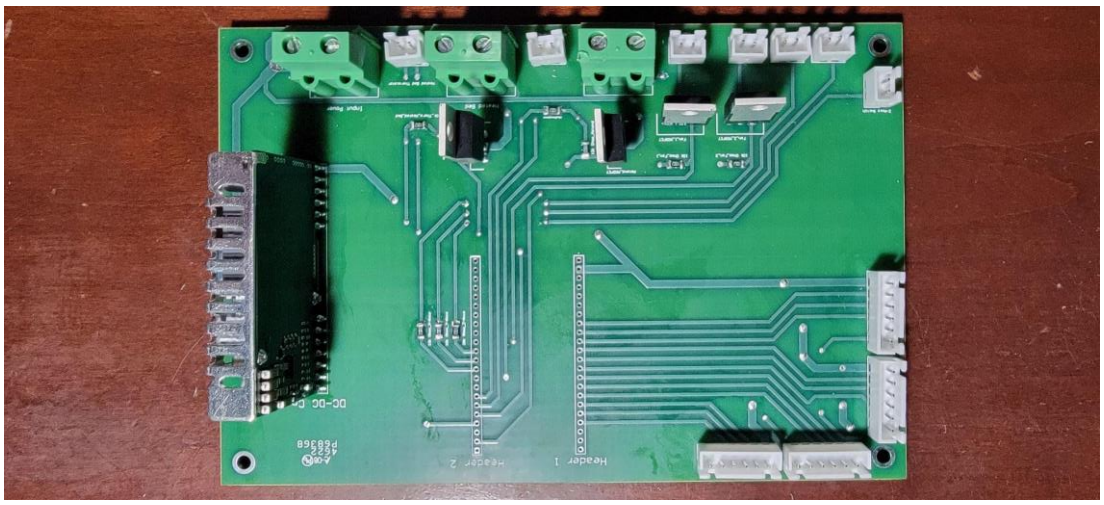


Figure 4. Main Control PCB

Item	Voltage	Current	Power
Heated Bed	23.68V	8.5A	201.28W
Hot Head	23.921V	1.7A	4.6657W
Fan 1	24.028V	64mA	1.6339
Fan 2	24.028V	47mA	1.1293W

Table 1. Main Control PCB validation results

The results were well within the constraints with the hot head pushing the limit but being only 1.8% percent over the expected current. The hot head can manage a small percentage and the circuit was designed for a potential 10% more current than expected.

The stepper motor controllers were not able to be more thoroughly evaluated due to issues within the stepper motor controller PCB. To validate the DC-DC converter could send the power to the stepper motor drivers in the future the output power for each driver was connected to a DC Load tester. The axis drivers were evaluated every 0.25A up to 1.25A while the extruder was tested every 0.25A and up to 1.5A. The results as seen below should work well with the stepper motor drivers with a maximum voltage loss of 0.2V.

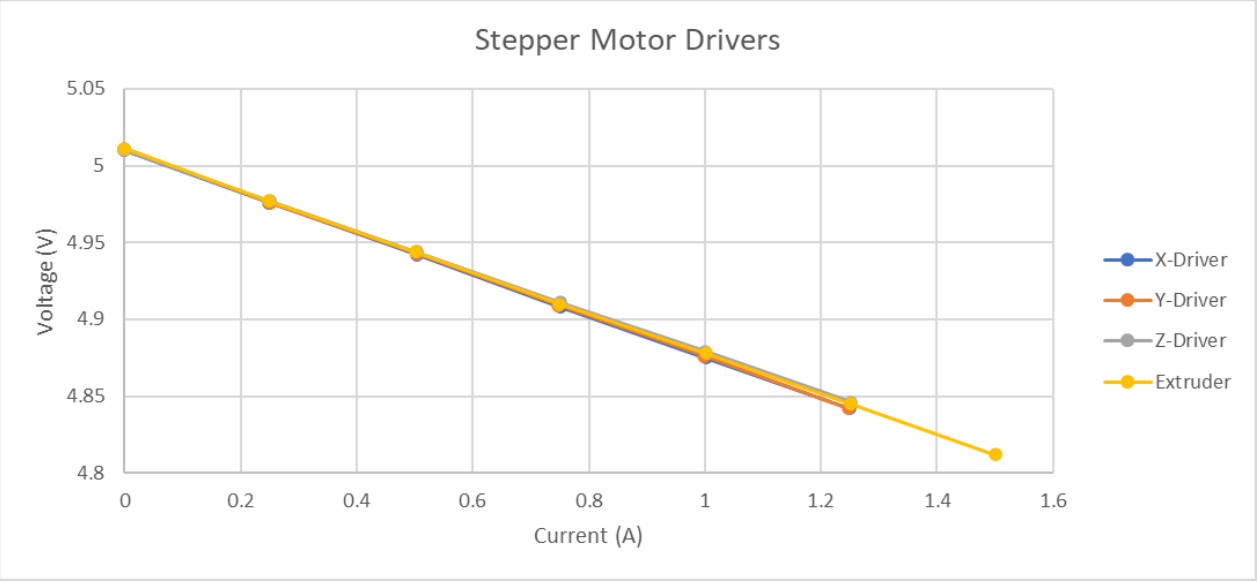


Figure 5. Load Testing DC-DC Converter

Amperage	Voltage	Power
0.000	5.010	0.000
0.249	4.976	1.239
0.503	4.942	2.486
0.752	4.908	3.691
1.002	4.875	4.885
1.250	4.842	6.053

Table 2. Load Testing the DC-DC Converter on the X-Driver Output

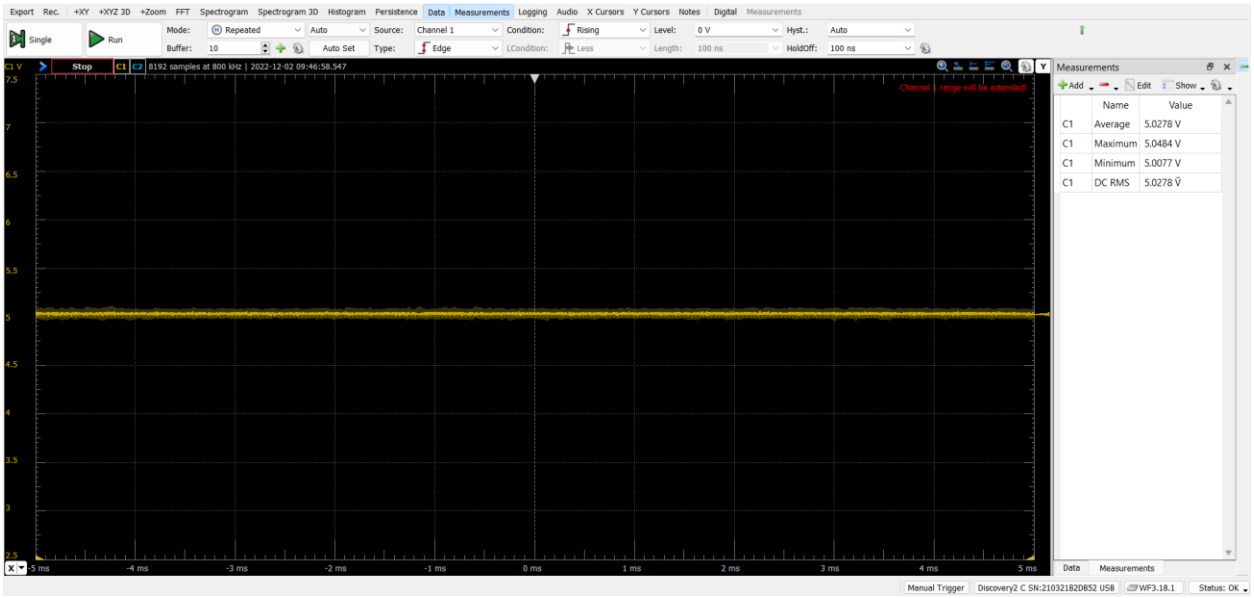


Figure 6. X-Driver at 0A Load

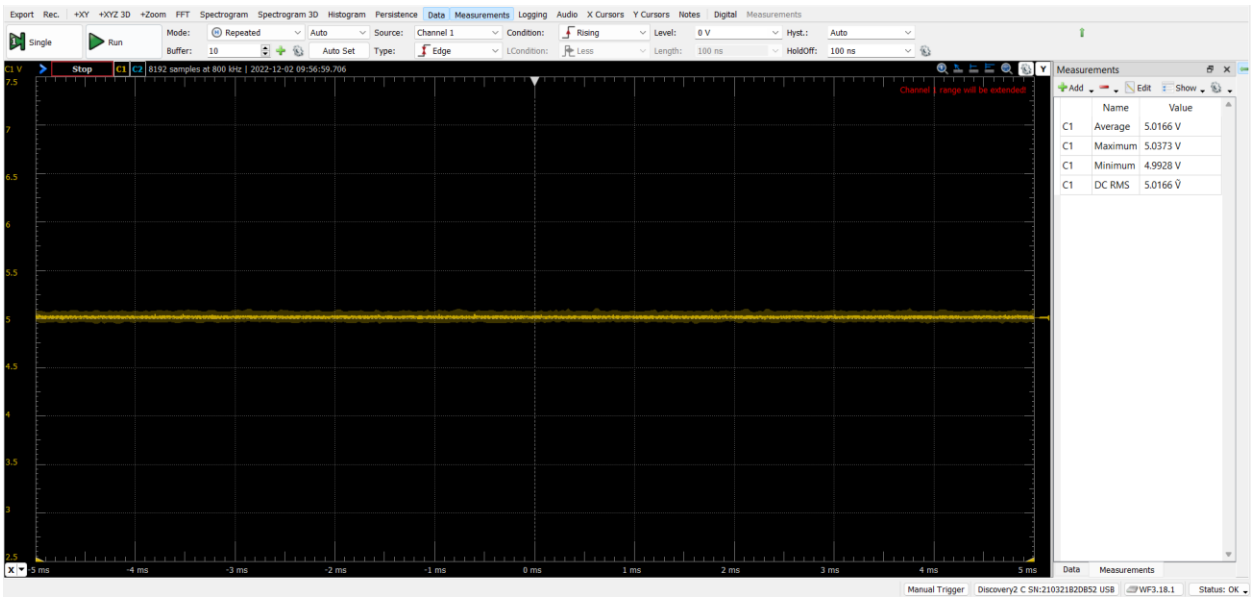


Figure 7. X-Driver at 0.249A Load

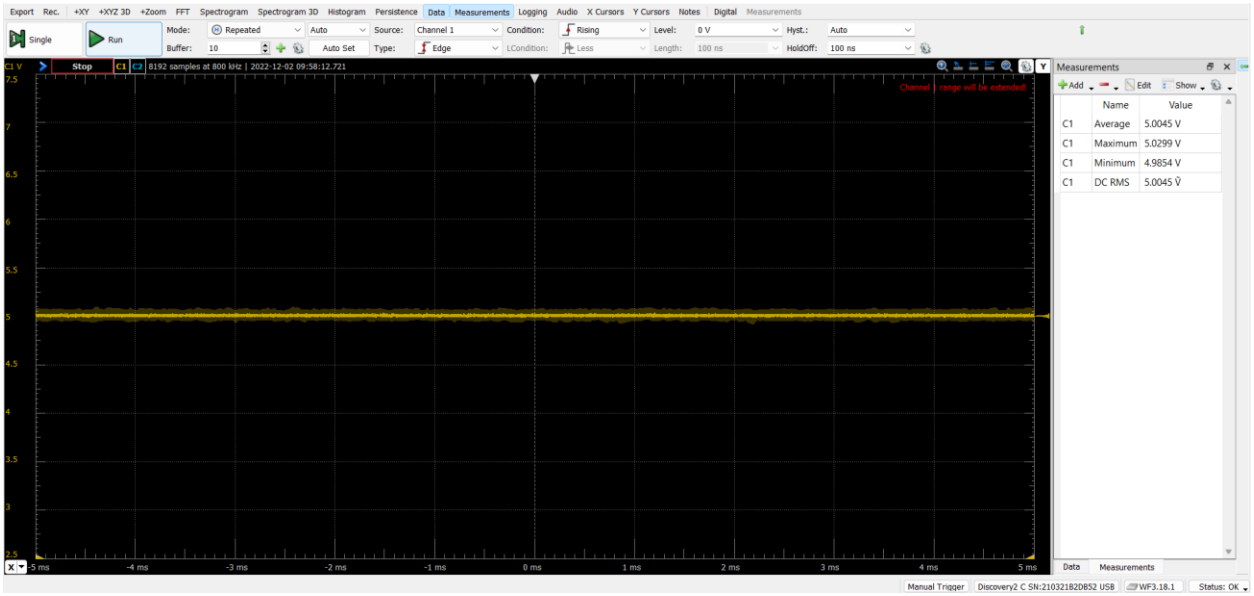


Figure 8. X-Driver at 0.503A Load

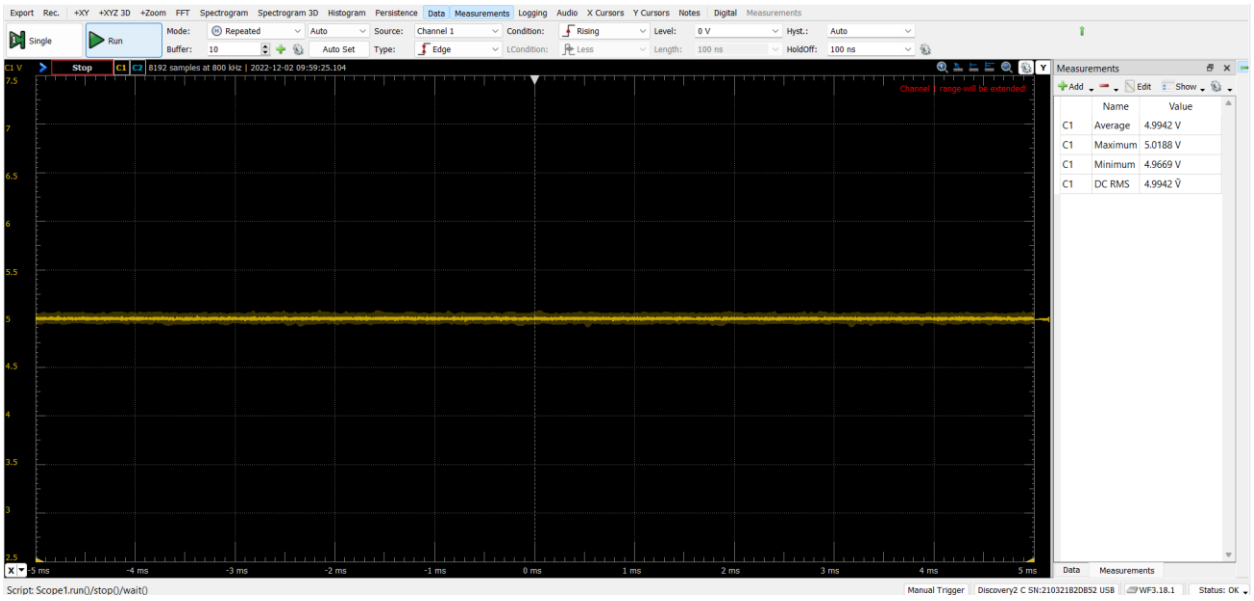


Figure 9. X-Driver at 0.752A Load

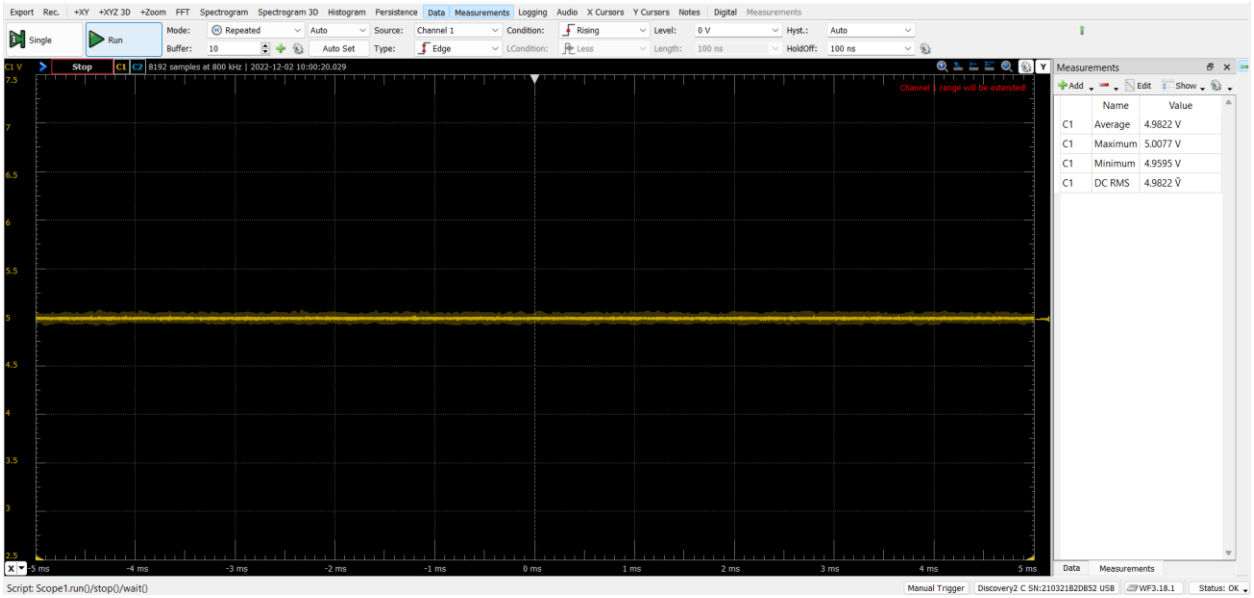


Figure 10. X-Driver at 1.002A Load

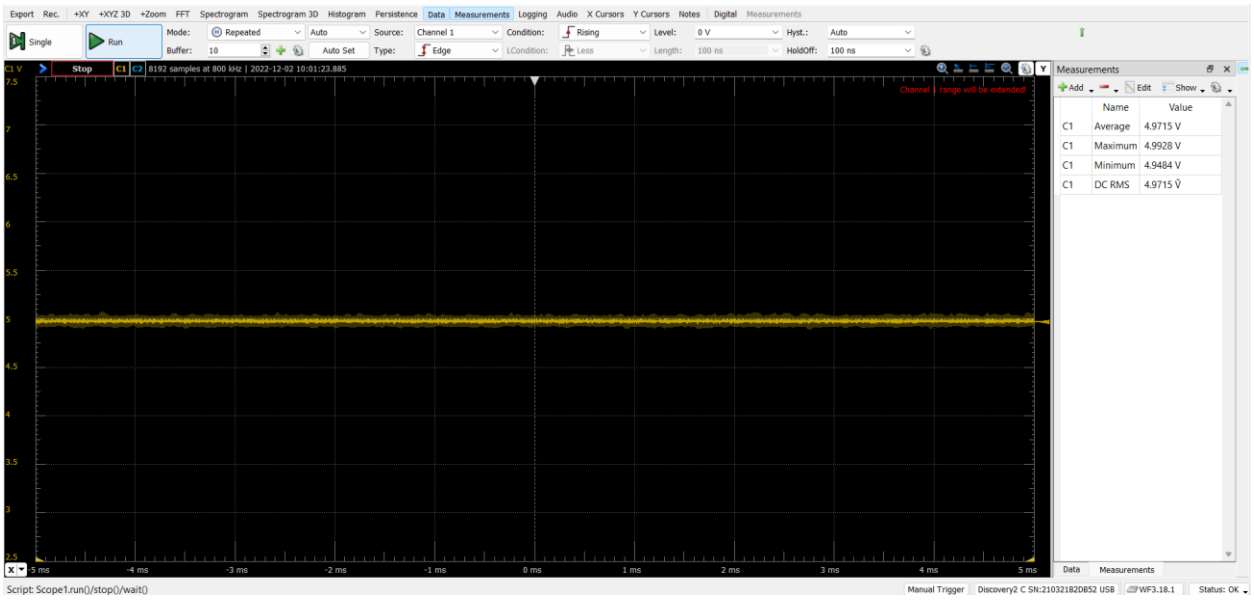


Figure 11. X-Driver at 1.250A Load

Amperage	Voltage	Power
0.000	5.011	0.000
0.250	4.976	1.244
0.503	4.943	2.486
0.748	4.909	3.672
0.999	4.876	4.871
1.248	4.842	6.043

Table 3. Load Testing the DC-DC Converter on the Y-Driver Output

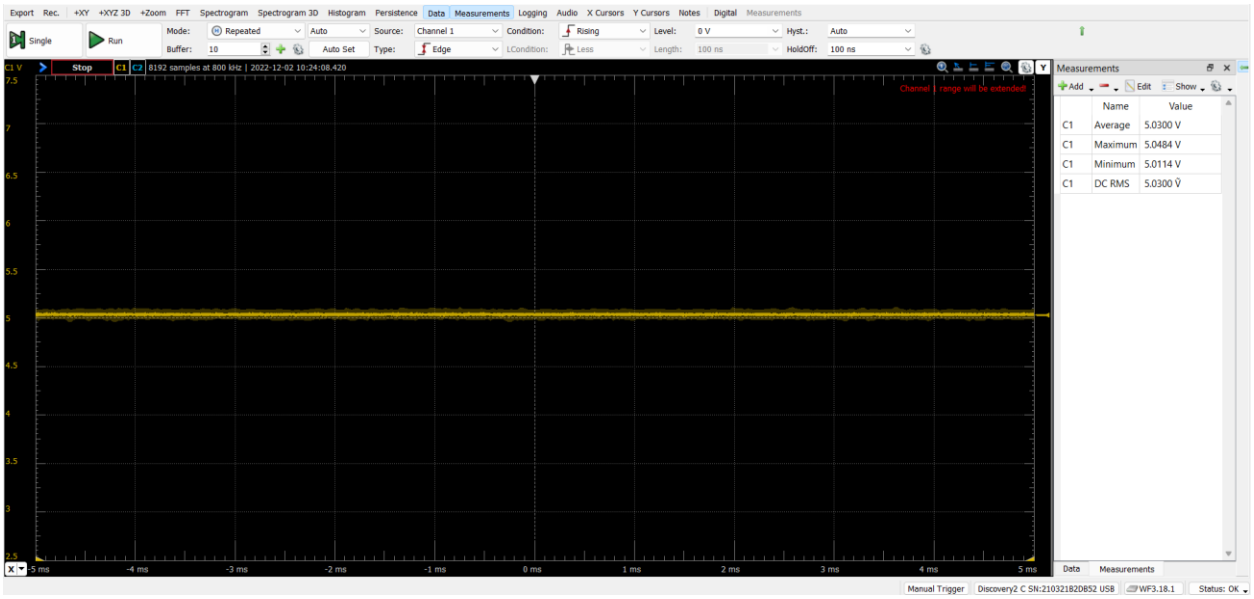


Figure 12. Y-Driver at 0A Load

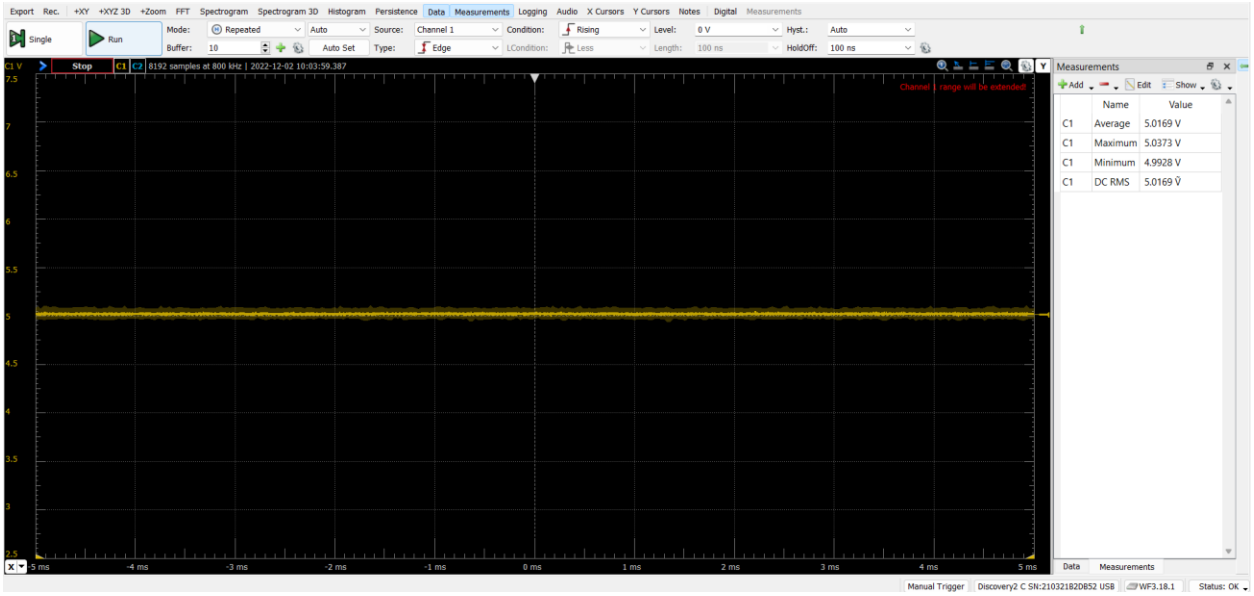


Figure 13. Y-Driver at 0.250A Load

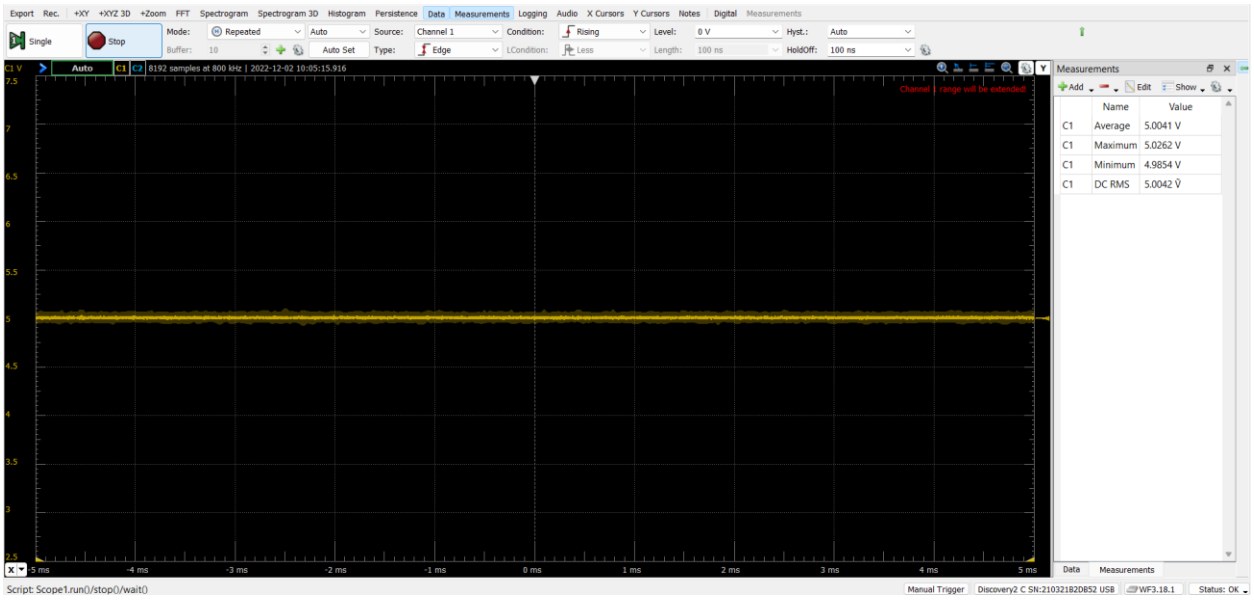


Figure 14. Y-Driver at 0.503A Load

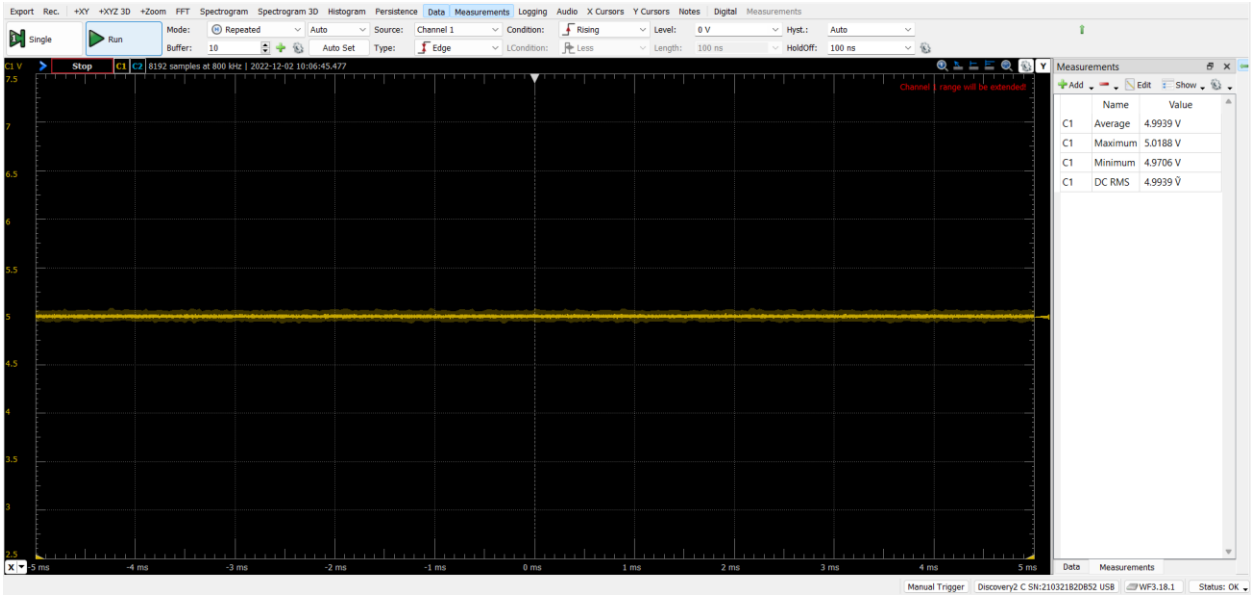


Figure 15. Y-Driver at 0.748A Load

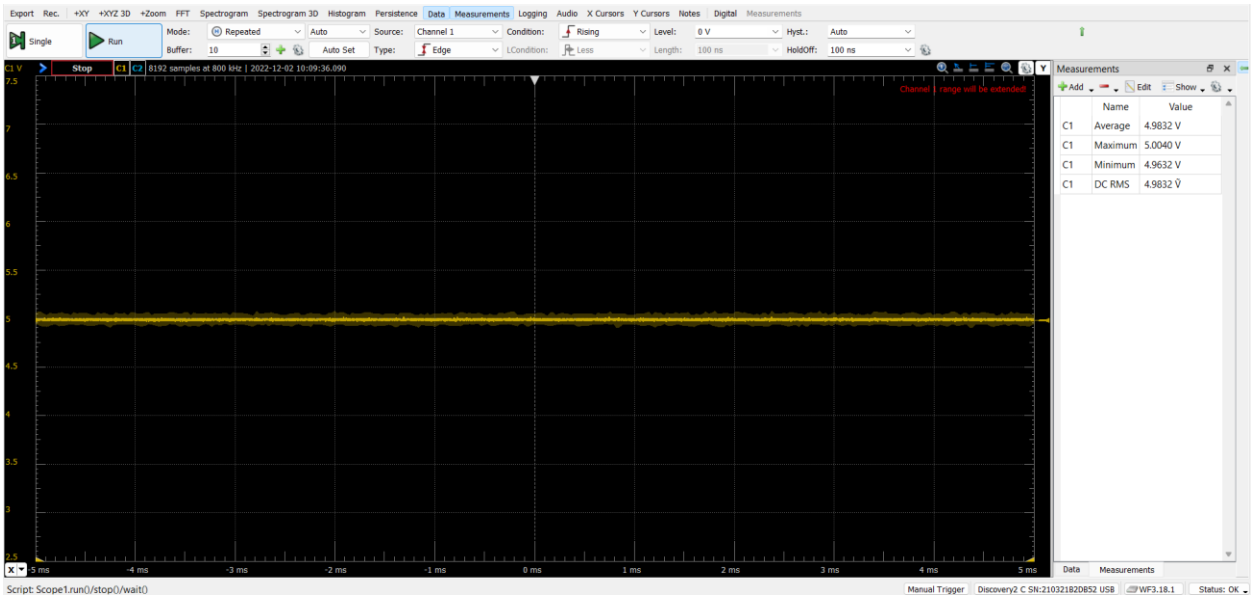


Figure 16. Y-Driver at 0.999A Load

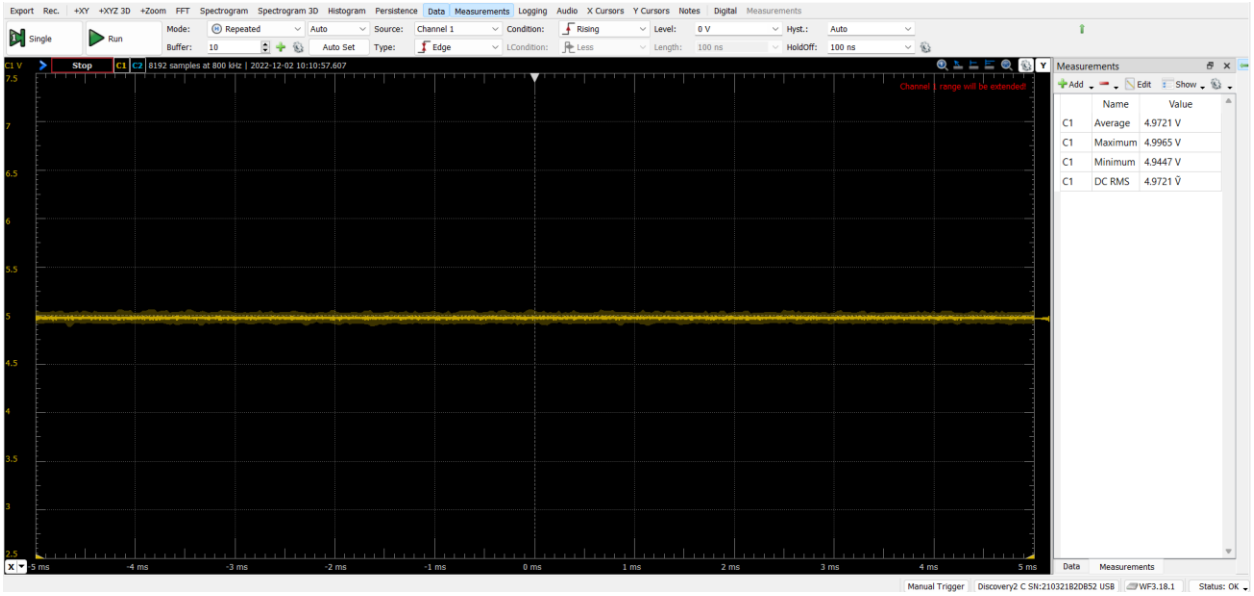


Figure 17. Y-Driver at 1.248A Load

Amperage	Voltage	Power
0.000	5.010	0.000
0.251	4.977	1.249
0.503	4.944	2.487
0.752	4.911	3.693
1.001	4.879	4.884
1.252	4.846	6.067

Table 4. Load Testing the DC-DC Converter on the Z-Driver Output

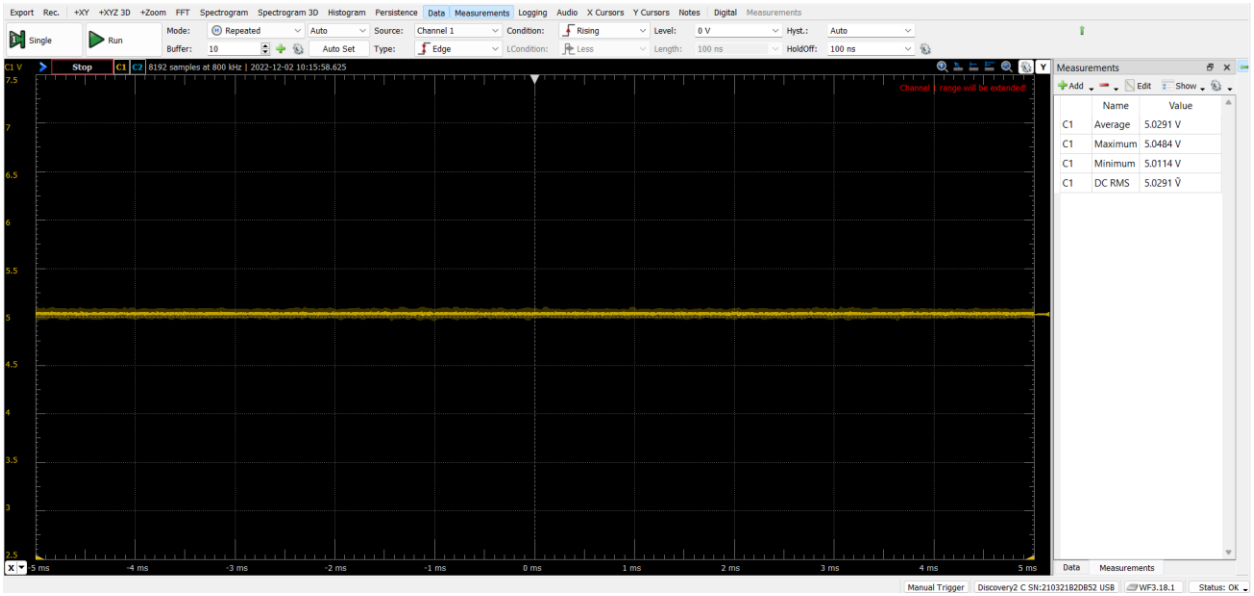


Figure 18. Z-Driver at 0A Load

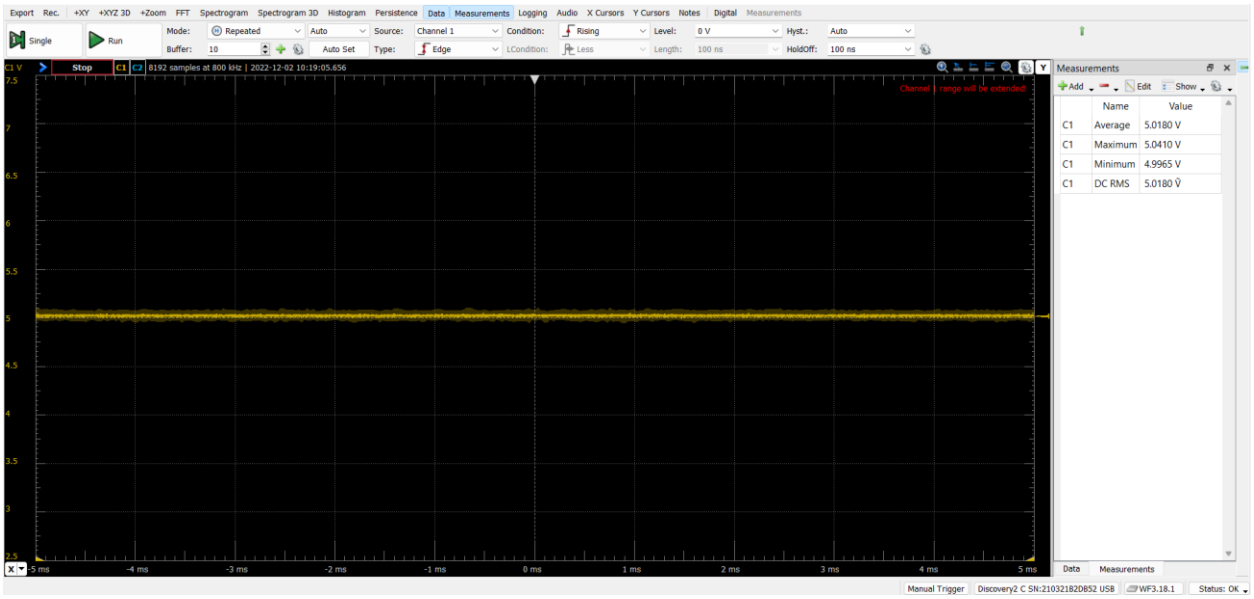


Figure 19. Z-Driver at 0.251A Load

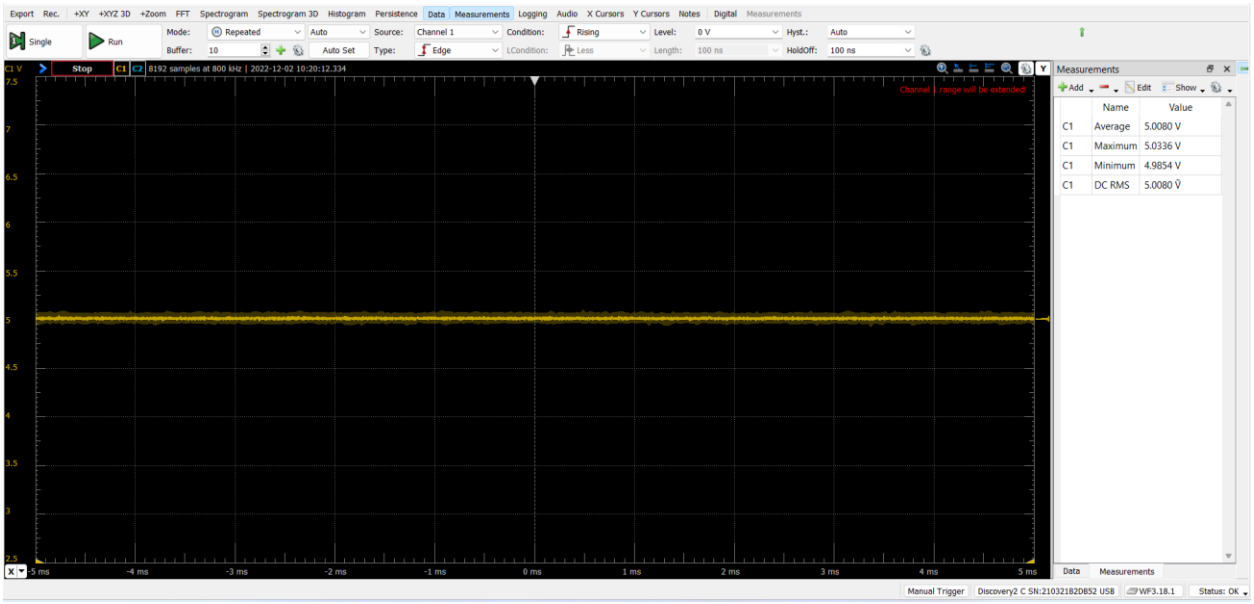


Figure 20. Z-Driver at 0.503A Load

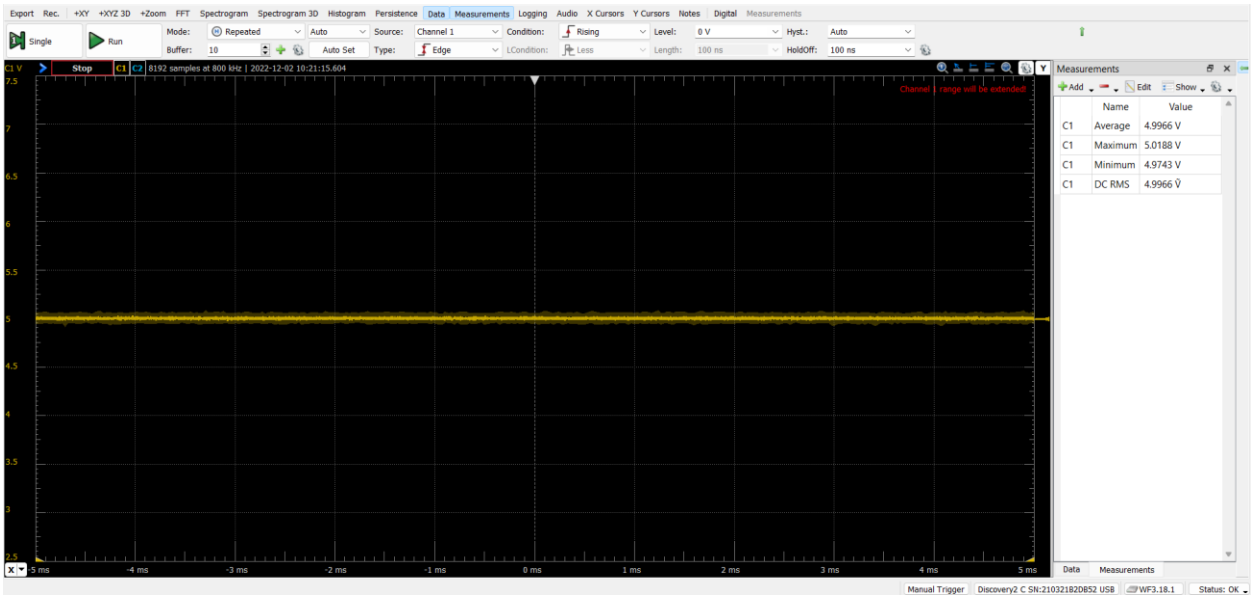


Figure 21. Z-Driver at 0.752A Load

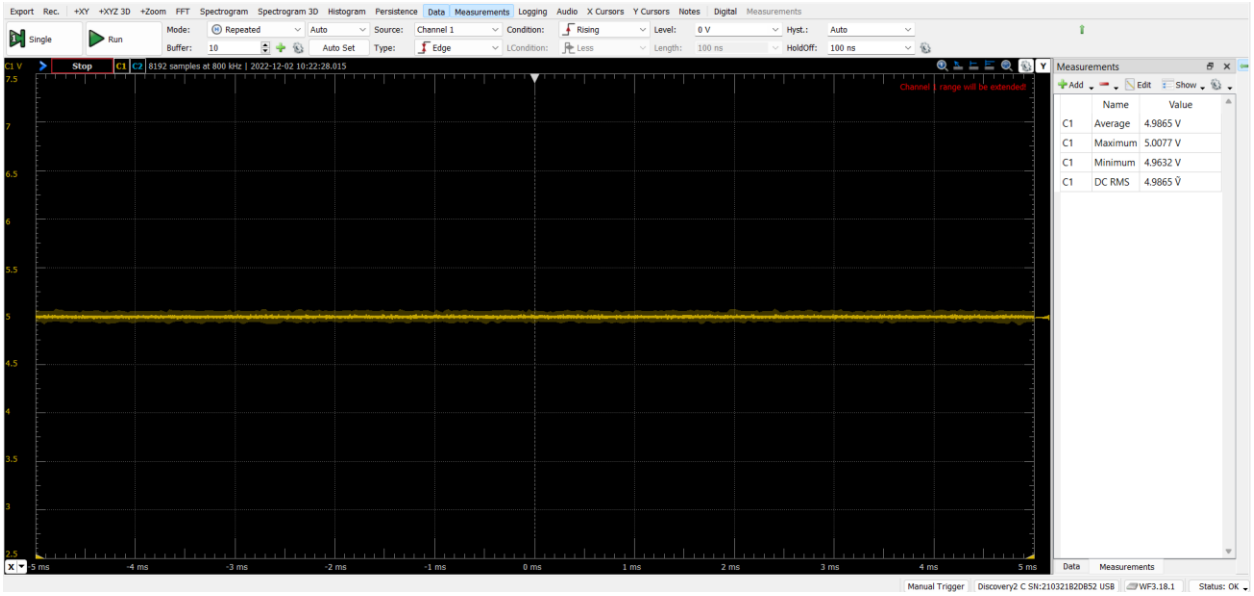


Figure 22. Z-Driver at 1.001A Load

Amperage	Voltage	Power
0.000	5.011	0.000
0.249	4.977	1.239
0.501	4.944	2.477
0.750	4.910	3.683
1.002	4.878	4.888
1.252	4.845	6.066
1.500	4.812	7.218

Table 5. Load Testing the DC-DC Converter on the Extruder Output

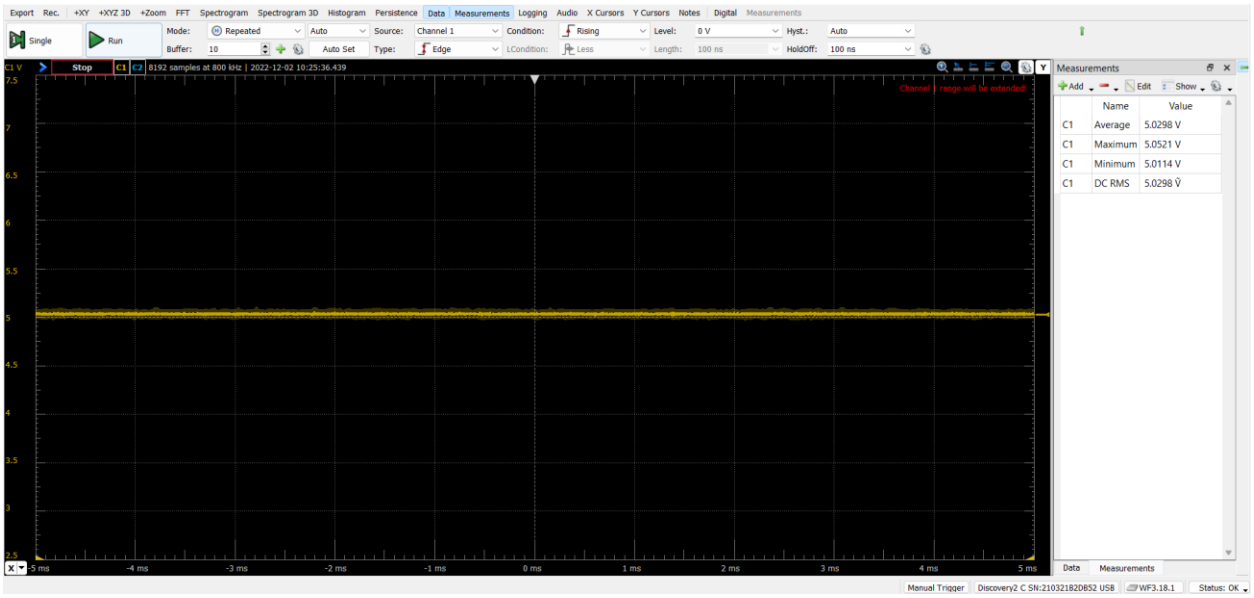


Figure 23. Extruder at 0A Load

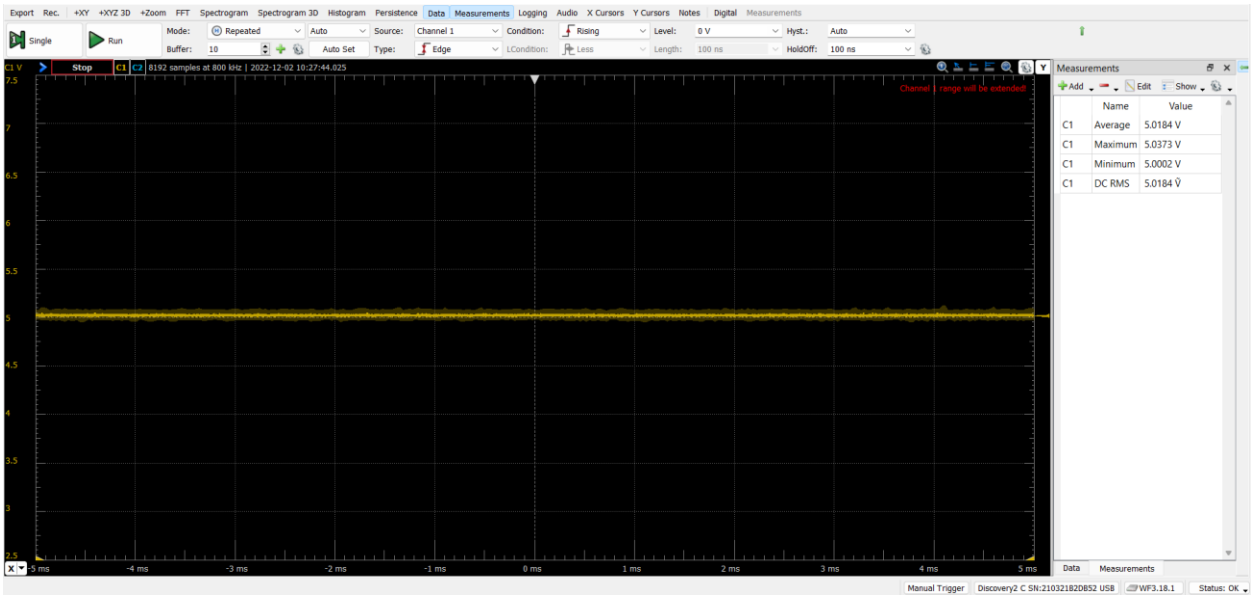


Figure 24. Extruder at 0.249A Load

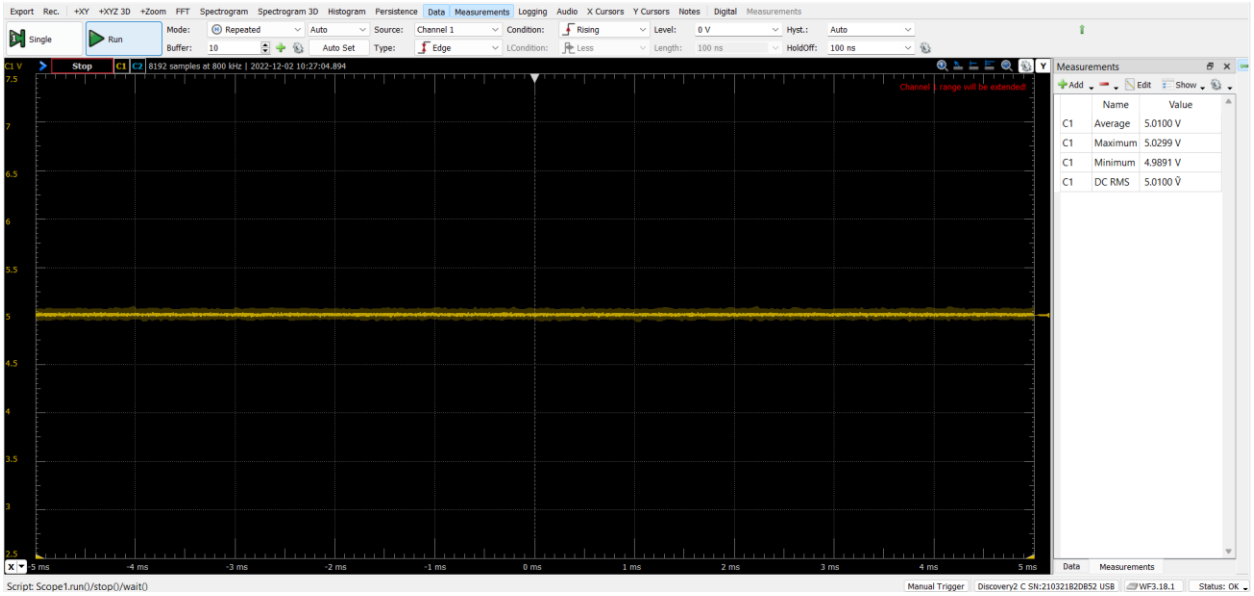


Figure 25. Extruder at 0.501A Load

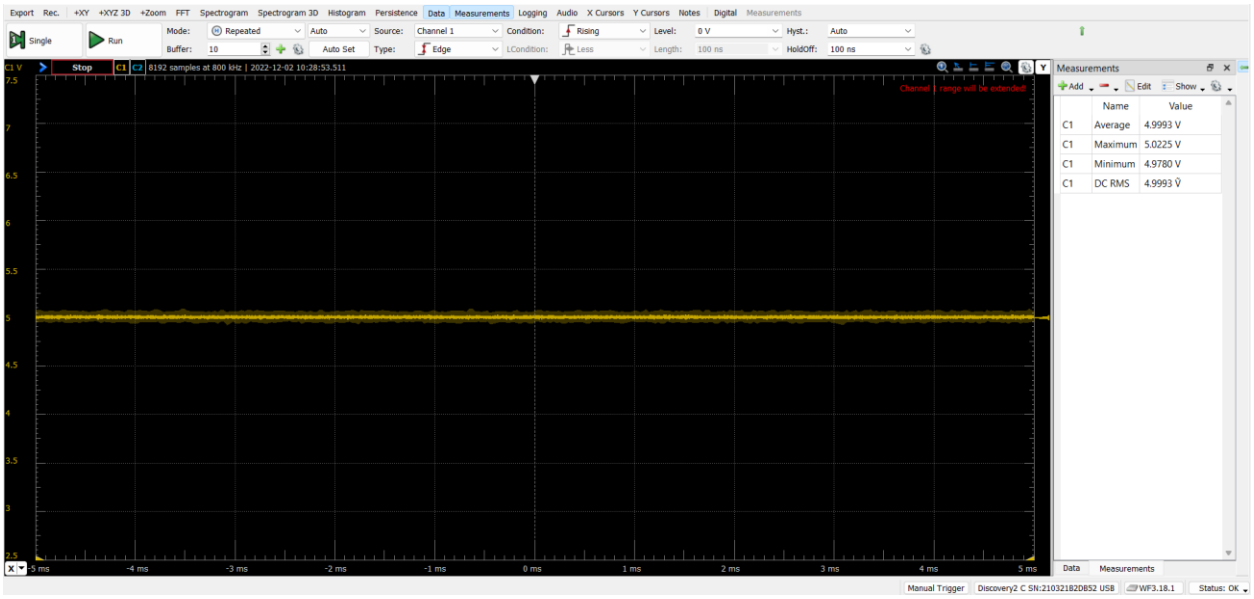


Figure 26. Extruder at 0.750A Load

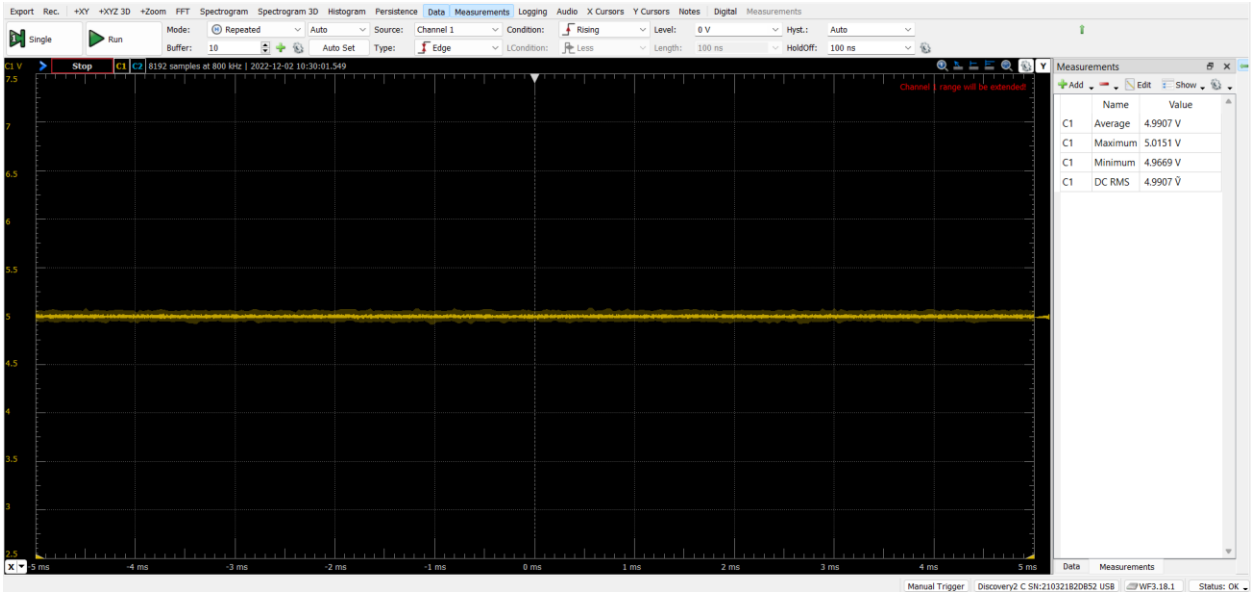


Figure 27. Extruder at 1.002A Load

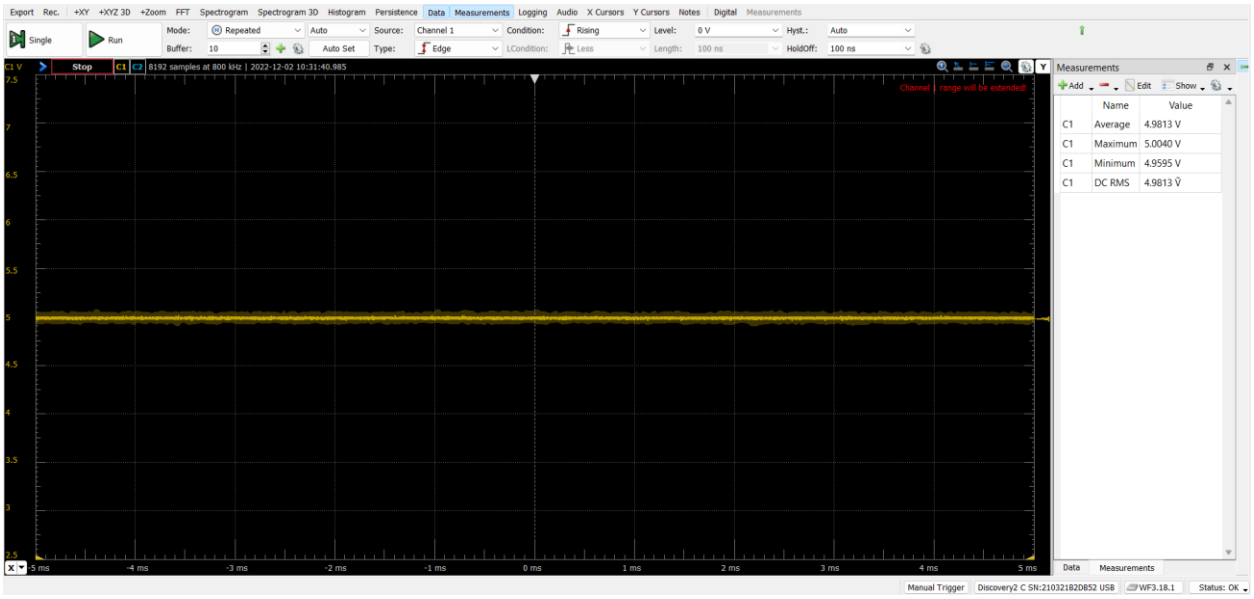


Figure 28. Extruder at 1.252A Load

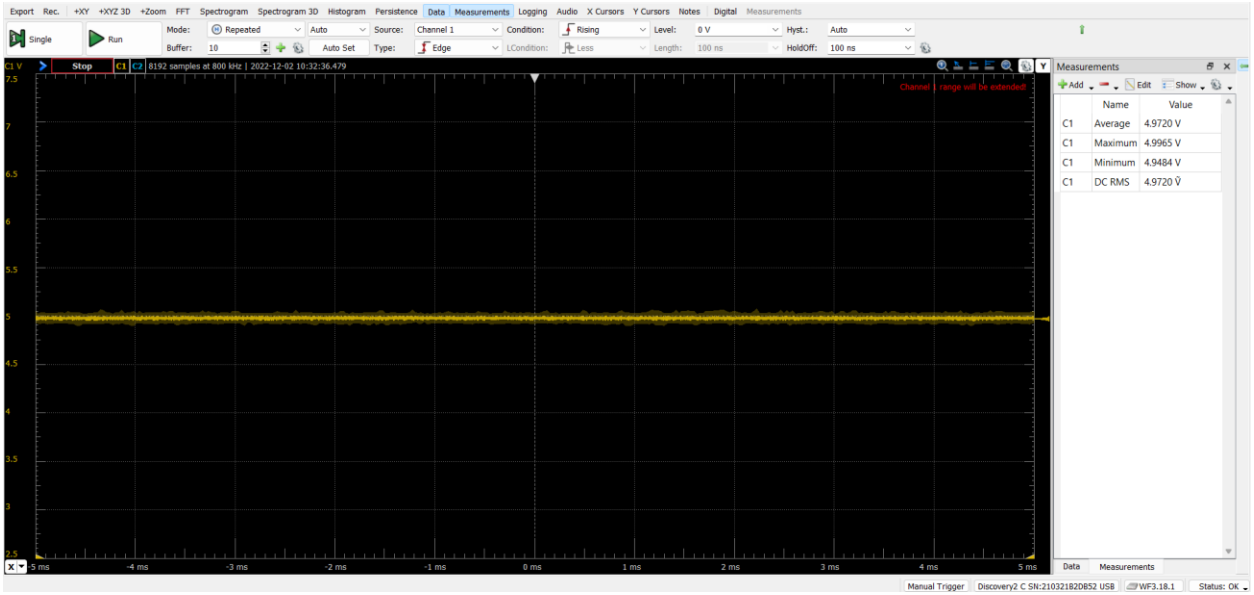


Figure 29. Extruder at 1.500A Load

2.3.2. Stepper Motor Controller PCB

The validation for the stepper motor controller PCB did not go as expected. The output connectors for the board were the wrong design so the footprint and via's were too small for the JST-XH male connectors. To be able to connect the board to the stepper motor, jumper wires were cut on one end and soldered into the via's, and the male end of the jumper wires were connected to the stepper motor connector. A special plug was then created that went between the main control board PCB and the stepper motor controller PCB. The connector would connect the ground and 5V power with JST-XH female plug and then having 3 jumper wires cut and attached to the stepper motor controller plug so that the AD2 could be

connected directly into the PCB. Once connected to the main control board, the stepper motor controller PCB receives power, ground, 3.3V 1kHz square wave, and 3.3V input as well as connected to the ground.

While the signals could be seen via an oscilloscope being sent to the TC-2130 motor driver, no signal was outputted to the stepper motors. Upon digging through the schematics, it was discovered that there connections missed in the design across all four stepper motor controllers. The connections missed were multiple grounds not connected to pins as needed and not connected to the TC-2130 chip ground pad. There was also an input voltage and circuitry missing on the B-side of the chip that was needed to send power to the stepper motors. The issues of the design, the stepper motor controller PCB could not be validated.

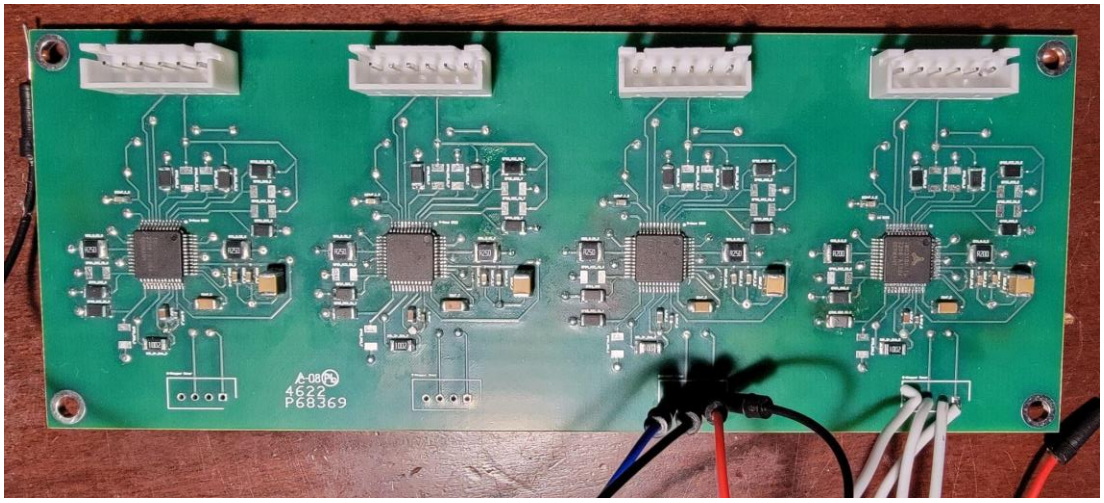


Figure 30. Stepper Motor Controller PCB

The plan going forward with the stepper motor controller is to redesign the circuit and have it verified and ready to be reprinted by next semester. The stepper motor controller schematic has been updated with the required connections. The schematics will have Professor Lusher review before moving forward with the PCB layout. Ideally the first week back in January the design should be sent out for printing and soldering to be completed the second week. The stepper motor controller PCB should have a successful validation the third week of the semester and be ready for integration.

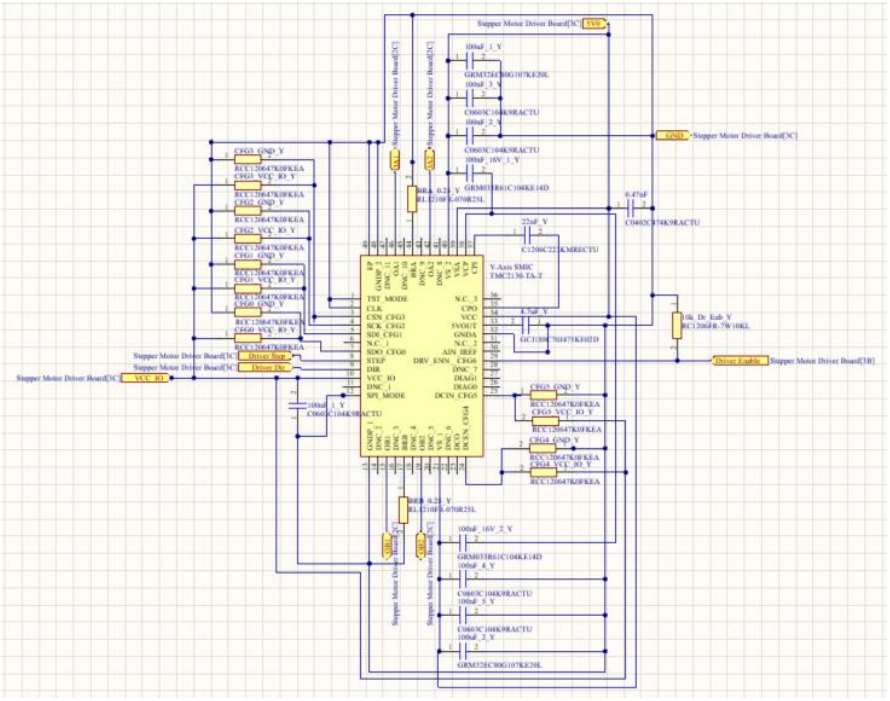


Figure 31. Updated Stepper Motor Controller Schematic

2.4. Printed Control Board Conclusion

The issues with the stepper motor controller PCB made validation of the board not possible. While unable to validate, the issues were discovered, and a plan was formulated to correct these issues. The main control PCB was partially validated as planned with only the MCU and stepper motor controllers being unable to be fully evaluated. The heated bed, hot head, and both fans were evaluated and validated as planned with no issues in those circuits.

3. Microcontroller Programming Report

3.1. Microcontroller Programming Introduction

The microcontroller unit subsystem consists of programming an ESP32-S3 to connect all subsystems together. It will control and communicate with all the printer components; multiple stepper motors, heated bed, hot head, extruder, fans, and web-hosted server.

3.2. Microcontroller Function Details

3.2.1. Connecting to Server

The MCU will connect to the server via local Wi-Fi network through the built-in Wi-Fi module. It will download files sent from the server and update the components accordingly. The MCU will store each line from the sent file into an array that will be compartmentalized for each update command. Every incoming line will be updated after execution of the current one until the whole file is read. As changes are made for each component, the updated information will be sent back to the server for the user to observe the printing progress through the phone application and alter print settings in realtime, if desired.

3.2.2. Converting G-code

The MCU code will take each line of G-code that is being read in and separate the commands to be sent to their respective output pins. Line commands beginning with "G" control the print head's linear motion, like XYZ-axis position and extrusion instructions. "G0" is used for non-extrusion movements while "G1" is for moves with extrusion. Line commands beginning with "M" control operations not involving movement, like fan speed and heating instructions. Parameters containing "X", "Y", or "Z" refer to the respective stepper motors. Parameters containing "E" pertain to the amount of filament being extruded through the hot head nozzle. Parameters containing "F" refer to the maximum movement rate between the start and end point of the current and subsequent moves until changed. This is known as the federate. Parameters with "S" refer to either speed or temperature commands. If any parameter is not updated for the next incoming G-code line, then the MCU will retain the previous setting for that parameter until it is changed or print job is stopped.

Component	Units
XYZE - Axis	mm
Feed Rate	mm/min
Temperature	Celsius
Fan Speed	0 (off) - 255 (on)

Table 6. Printer Components and their Units

3.2.3. Controlling Components

The MCU's pins will be driving all the components. The X and Z-axes move along with the hot head while the Y-axis moves along with the heated bed. The E-axis, or extrude feeder, moves through the hot head's nozzle. The feed rate determines the speed at which all these axes operate. The speed of the fan and the temperatures of the heated components are updated whenever an "S" parameter is executed. Stepper motors are controlled by calculating the number of micro steps it takes to move the print head to the desired location on the print bed. The accuracy of such movement needs to be within the hundred-thousandths place of a mm. Once conversion is determined, the pins connecting to these components will send on/off pulse signals to activate them. The pins responsible for controlling the heated components will output a voltage proportional to the necessary temperature until the requirement is reached. The changeable fan speed will be controlled by sending an on/off signal to its pin for the specified setting.

3.3. Microcontroller Program Validation

The MCU programming was validated through executing the written code in the Arduino IDE on a breadboard. The successful functionality of the ESP32-S3 pins used for driving the stepper motors, fans, and heated components was indicated by programming LEDs to turn on and off in a successive order.

```
switch(command) {
  case 1:   //linear move
      FEEDRATE = F * STEPS/*step amt (mm/min)*/;
      MOTOR_X = X * STEPS/*step amt (mm)*/;
      MOTOR_Y = Y * STEPS/*step amt (mm)*/;
      MOTOR_Z = Z * STEPS/*step amt (mm)*/;
      EXTRUDE_AMT = E * STEPS/*step amt (mm)*/;
      break;
    case 28:  //auto home
      /*special parameters*/
      MOTOR_X = 0;
      MOTOR_Y = 0;
      MOTOR_Z = 0;
      EXTRUDER = 0;
      break;
    case 90:  //absolute positioning
      //in reference to original home
      MOTOR_X = X;
      MOTOR_Y = Y;
      MOTOR_Z = Z;
      break;
    case 91:  //relative positioning
      //in reference to the previous command line
      MOTOR_X = X + X;
      MOTOR_Y = Y + Y;
      MOTOR_Z = Z + Z;
      break;
```

Figure 32. Snippet of Code: Case Statements for G-code Commands

The Wi-Fi module proved to be working correctly by connecting to a local hotspot for internet connectivity. Communication to the Firebase server then proved to be successful by updating and outputting the XYZ-axes and Extruder, heated bed and hot head temperatures, and fan speed in real time.

```
/*
connecting ESP32-S3 to Firebase through WiFi
*/
#include <WiFi.h>
#include <FirebaseESP32.h>

#define FIREBASE_HOST "https://d-printer-b61f0-default-rtdb.firebaseio.com/"
#define FIREBASE_AUTH "N9Sx45ahrrGiOtH4DnJRjGf6wurleIxwdpEJJRRT"
#define WIFI_SSID "Moto Z"
#define WIFI_PASSWORD "esp32pass"

//Firebase data object defined
FirebaseData firebaseData;
FirebaseJson json;
```

Figure 33. Snippet of Code: Firebase and Wi-Fi Connection

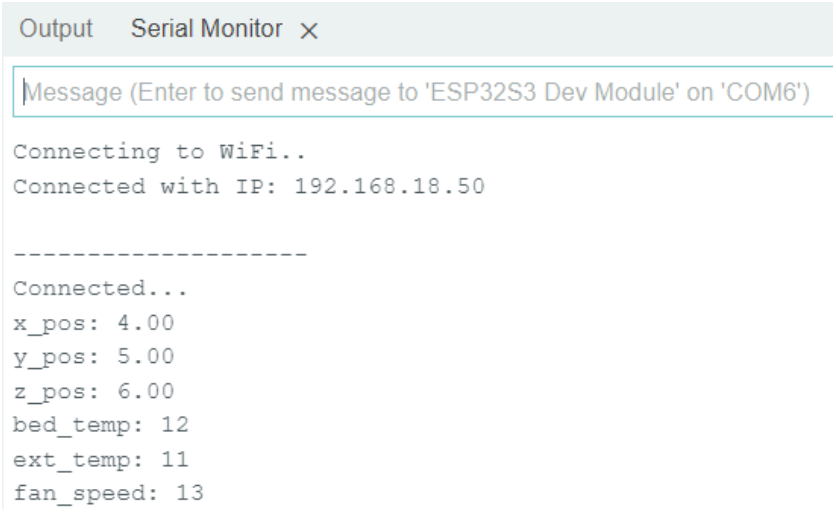


Figure 34. Wi-Fi Connection Verification and Output of Parameters in Firebase

The final demonstration showed how a mock G-code script input from the serial monitor was stored and compartmentalized into an array. Doing so allows each command and its following parameters to correctly update their respective components in the proper order. The true test of full functionality will be realized when all the subsystems are officially combined. The selected pins from this demonstration may be altered to match that of which is on the PCB. The separate codes of this demonstration will be combined into one seamless program in the Espressif IDE.

```
ESP-ROM:esp32s3-20210327
G1 X128.639 Y120.475 E412.06085  G0 F9000 X128.508 Y120.938  M105  M190 S50  G1 F2700 E412.06635  G90

0: "G1"
1: "X128.639"
2: "Y120.475"
3: "E412.06085"
4: ""
5: "G0"
6: "F9000"
7: "X128.508"
8: "Y120.938"
9: ""
10: "M105"
11: ""
12: "M190"
13: "S50"
14: ""
15: "G1"
16: "F2700"
17: "E412.06635"
18: ""
19: "G90"
20: ""
```

Figure 35. Output of Sample G-code Stored in Array

3.4. Microcontroller Programming Conclusion

The microcontroller subsystem performs correctly and will provide full support for the other subsystems to communicate properly with each other in one complete system. Programming the microcontroller was an in-depth learning experience. Understanding through Marlin Firmware how a 3D printer implements G-code was interesting. Writing the code for this subsystem in the Arduino IDE made understanding what the code is supposed to do much easier. Rewriting the code for the ESP32-S3 in the correct IDE will be more coherent now.

4. Phone Application and Web-Based Server Report

4.1. Application and Server Introduction

The application and server will provide the graphical user interface for processing and communicating user-enabled features to the 3D printer. The application will use three different tabs, home, library, and formatting tabs, to carry out all features of the original 3D printer. The application will use the Firebase server to store files and implement the library of print files, including related file information. The Firebase server will also be the intermediary of communications between the application and the physical printer using the Firebase Realtime Database.

4.2. Application Details

4.2.1. Home Tab

The home tab incorporates two key features of the printer. The first is displaying the current fan speed, position of the extruder, and temperatures of the bed and extruder. Using Firebase-Kotlin function calls, the values displayed to the user will update upon any change to the values stored in the FRD (Firebase Realtime Database). The second feature is the start and stop print commands. To send a start command, the user first needs to select a print file from the library tab. Otherwise, the application will send a message informing the user to select a print file. Only after sending the start command will the user be able to stop the print by holding down the same button. When the user has selected a file from the library tab and pushed the button to send the start command, two values in the FRD will be updated to provide the MCU with the necessary information to start a print. The “beginPrint” value will hold the string “true,” and the “URL” value will hold the URL to the print file in FS (Firebase Storage) which the MCU can use to download the print file from the FS. If the user chooses to cancel the print by holding down the print button, these values will update to “false” and “null” respectively where “null” will remove the “URL” value in the FRD until a new one is provided.

4.2.2. Library Tab

The library tab works heavily with FS and FRD to implement the interactive library where users can upload new print files, view them in the library window and select one to print. After pushing the upload print file button, the application will first ask the user to check for existing G-code files in their mobile device before proceeding. If the user chooses to continue, the application will launch an activity into the devices folder and pull the URI (uniform resource identifier) of the file selected by the user. To complete the sequence of user events for uploading a print file, the user will be prompted to enter a file name. Behind the UI, the application will first use the URI and Firebase-Kotlin function calls to send the print file to Firebase storage and pull the URL download link from its location in the FS. The application will then produce a gcodeClass data object to be sent to the FRD containing the name, size, timestamp, and FS URL of the print file. Upon updating the “Print Files” path in FRD with the new gcodeClass data object, an adapter class and a Kotlin visual resource tool will create the library view where users can see the related file information mentioned above and select the file they wish to print. Once selected, the file name and URL download link will be

communicated to the home tab using Kotlin fragment transactions. The file name will be displayed to the user and the URL will be held in a variable ready to be sent to the FRD upon pushing the “Start Print” button.

4.2.3. Formatting Tab

The formatting tab implements the original 3D printer’s feature of moving the extruder to custom positions as well as adjusting fan speed and setting heating temperatures for both the bed and extruder. The bed and extruder temperature and extruder position fields use range filters to prevent users from entering values too high or low. The fan speed has a three-state switch for each possible mode: “OFF”, “HIGH”, or “LOW.” Once users have made their inputs or changes and pushed the apply formatting button, the “Printer Formatting” path in the FRD will update with the user's custom formats.

4.3. Server Details

4.3.1. Firebase Storage

The Firebase storage will house all print files removing the need to store print files on the device. The FS will only be accessible through the function calls within the application itself. Neither the user nor MCU will have access to update or edit the print file storage. To access the files in FS, the application will use Firebase-Kotlin function calls to add files and retrieve the URL download link which will be placed in a gcodeClass data object to be shared to the library or home tab upon selection.

4.3.2. Firebase Realtime Database

The Firebase Realtime Database acts as the intermediary between the application and MCU while also managing data to display in the library tab. The FRD has three key paths: Print Files, Printer Formatting, and Start Print. The Print Files hold G-code files uploaded by the user along with related file information such as the name, size, timestamp, and URL. The application accesses the FRD through Firebase-Kotlin function calls which can be similarly performed by the MCU to receive or transmit the information from Firebase. The second key path is the Printer Formatting path where users can select custom formats and apply them to the physical printer. The Start Print key path commands the physical printer to begin printing and provides the download link for the MCU.

4.4. Application Programming

Due to issues between Android Studio and Github the master branch of the repository is not updated. However, the provided link is the current working directory: https://github.com/liusteven2/3D_Printer_1/commit/77b39dc8cc0bdacb78247d19cec3773e36c29269. Below are a few key blocks of code from each tab of the application.

4.4.1. Home Tab Communicating Print Initiation to FRD

Figure 36. shows the block of code triggered on the “Start Button” click. In this block of code, Firebase-Kotlin function calls are used to set values under the “Command Print” path in order to begin printing.

```
btn.setOnClickListener{  it: View!
    fileUrl = args?.get("url").toString()
    database = FirebaseDatabase.getInstance().getReference( path: "Start Print")
    val commencePrint = BeginPrint( beginPrint: "true",fileUrl)
    database.child( pathString: "Command Print").setValue(commencePrint).addOnSuccessListener {  it: Void!
        Toast.makeText(activity,  text: "Begin Print!", Toast.LENGTH_LONG).show();
    }.addOnFailureListener {  it: Exception
        Toast.makeText(activity,  text: "Begin Print Failed", Toast.LENGTH_SHORT).show();
    }
    btn.text = "Printing! Hold to cancel print."
    btn.setTextSize(TypedValue.COMPLEX_UNIT_SP,  size: 20F)
}
```

Figure 36. Communicating Print Initiation to FRD

4.4.2. Home Tab Retrieving Formatting Values

Figure 37. shows the block of code performing Firebase-Kotlin function calls for retrieving data. This block is pulling data in the “Printer Formatting/Format” path which corresponds to the formatting values.

```
private fun getUserData() {
    databasePC = FirebaseDatabase.getInstance().getReference().child( pathString: "Printer Formatting").child( pathString: "Format")
    val postListener = object : ValueEventListener {
        override fun onDataChange(snapshot: DataSnapshot) {
            val pc = snapshot.getValue<PrinterControls>()
            bedTempDisplay?.setText(pc?.bed_temp.toString()+"\u00B0")
            extTempDisplay?.setText(pc?.ext_temp.toString()+"\u00B0")
            extPosDisplay?.setText(pc?.x_pos.toString()+","+pc?.y_pos.toString()+","+pc?.z_pos.toString())
            fanSpeedDisplay?.setText(pc?.fan_speed.toString())
            simpleChronometer?.start()
        }

        override fun onCancelled(error: DatabaseError) {
            Toast.makeText(activity,  text: "DID NOT FIND DATA", Toast.LENGTH_SHORT).show();
        }
    }
    databasePC.addValueEventListener(postListener)
}
```

Figure 37. Retrieving Formatting Values from FRD

4.4.3. Library Tab Launching File Chooser Activity

Figure 38. shows the block of code used to retrieve print files on the user’s device. The code below implements a pop-up view to ask the user for a file name and whether to accept this file to upload or cancel.

```
val getFile = registerForActivityResult(ActivityResultContracts.GetContent(),
    ActivityResultCallback {  it: Uri?
        if (it != Uri.EMPTY) {
            fileUri = it!!
            val builder = AlertDialog.Builder(requireActivity())
            val editText_view =
                inflater.inflate(R.layout.edit_text_layout, container,  attachToRoot: false)
            val editText: EditText = editText_view.findViewById(R.id.et_editText)

            with(builder) {  this: AlertDialog.Builder
                setTitle("Enter name of file!")
                setPositiveButton( text: "OK") { dialog, which ->
                    fileName = editText.text.toString()
                    uploadFile()
                }
                setNegativeButton( text: "Cancel") { dialog, which ->
                    Toast.makeText(activity,  text: "Canceled File Upload", Toast.LENGTH_SHORT)
                        .show();
                }
                setView(editText_view)
                show()  ^with
            }
        }
    })
```

Figure 38. Launching Activity for Choosing Device Files

4.4.4. Library Tab Uploading Function

Figure 39. The block of code below is a portion of the uploadFile function created to upload the user-chosen print file to the Firebase Storage, retrieved its URL, and upload the file and related information to Firebase Realtime Database. Below progressDialog function is used to provide users a visual loading image while the application is performing the upload to FS and FRD. The last visible line is the gcodeClass data object being created with all related file information before being uploaded to FRD.

```
private fun uploadFile() {
    val progressDialog = ProgressDialog(activity)
    progressDialog.setMessage("Uploading File...")
    progressDialog.setCancelable(false)
    progressDialog.show()

    val formatter = SimpleDateFormat( pattern: "yyyy_MM_dd_HH_mm_ss", Locale.getDefault())
    val now = Date()
    fileNameNow = formatter.format(now)
    storage = FirebaseStorage.getInstance().getReference( location: "Print Files/"+fileName)
    uploadTask = storage.putFile(fileUri!!)
    var pfd = requireActivity().contentResolver.openFileDescriptor(fileUri!!, mode: "r")
    var fileLength: Long? = pfd!!.getStatSize()
    fileLengthReadable = fileLength?.readableFormat()


    uploadTask.addOnFailureListener {  it: Exception
        Toast.makeText(activity,  text: "File Upload to Storage Failed", Toast.LENGTH_SHORT).show()
        if (progressDialog.isShowing) progressDialog.dismiss()
    }.addOnSuccessListener{  it: UploadTask.TaskSnapshot!
        Toast.makeText(activity,  text: "File Upload to Storage Success", Toast.LENGTH_SHORT).show()

        storage.downloadUrl.addOnSuccessListener {  it: Uri!
            fileUrl = it.toString()
        }

        gcodeFile = gcodeFileClass(fileName, fileUrl, fileNameNow.toString(), fileLengthReadable.toString())
```

Figure 39. Uploading File to Firebase Storage

4.4.5. Formatting Tab Uploading New Formatting Values

Figure 40. shows the block of code used to upload new formatting values to FRD. Upon button click, a database reference is used to first get the correct path within FRD. The following line creates a PrinterControls object storing each user input value. The next lines perform an update to the values within the FRD and provide messages to the user upon success or failure.

```
btn.setOnClickListener{  it: View!
    database = FirebaseDatabase.getInstance().getReference( path: "Printer Formatting")
    val newFormat = PrinterControls(extTemp.text.toString(), bedTemp.text.toString(), fanSpeed.text.toString(), xVal.text.toString(), yVal.text.toString(), zVal.text.toS
    database.child( pathString: "Format").setValue(newFormat).addOnSuccessListener {  it: Void!
        Toast.makeText(activity,  text: "Successfully Saved" + extTemp.text.toString(), Toast.LENGTH_SHORT).show();
    }.addOnFailureListener {  it: Exception
        Toast.makeText(activity,  text: "Failed Saved", Toast.LENGTH_SHORT).show();
    }
}
```

Figure 40. Uploading Formatting Values

4.5. Validation

4.5.1. Library Tab, Firebase Storage, and Firebase Realtime Database Validation

The following figures 41-44 show a working library tab in conjunction with a working Firebase storage and Firebase realtime database. The figures below demonstrate FS and FRD being updated when a user has selected a file to upload. The user names the print file “MAROON” which is then added to the FS. From the FS, the URL is taken and added to the

FRD along with related file information. The library tab then produces a new print file option “MAROON” with the related file information stored in the FRD.

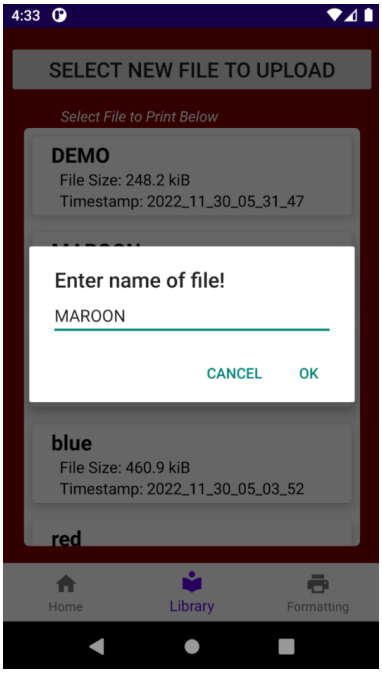


Figure 41. File Naming “MAROON”

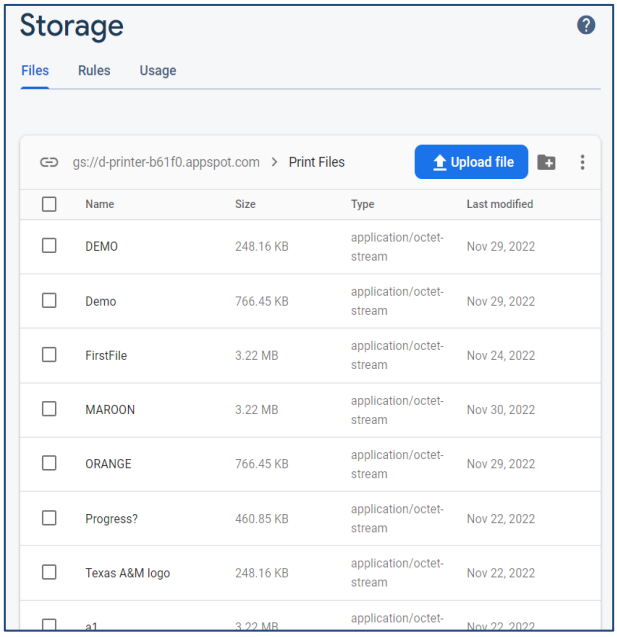


Figure 42. File Stored in Firebase Storage

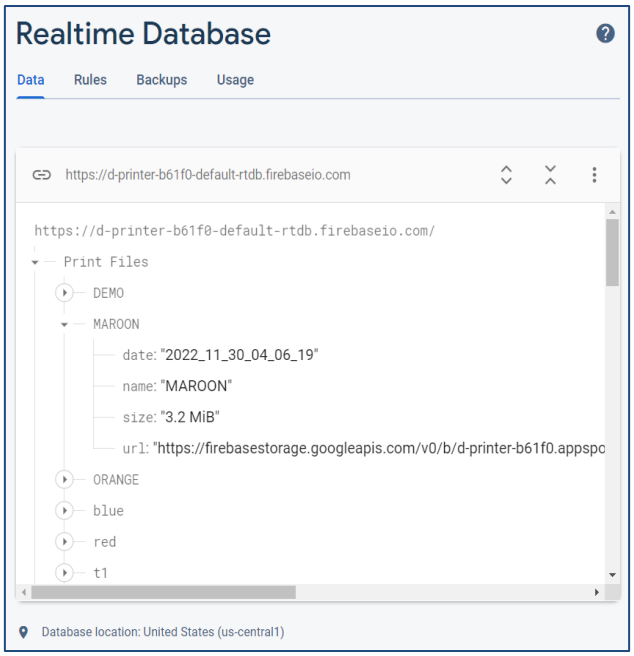


Figure 43. File Information Stored in Firebase Realtime Database

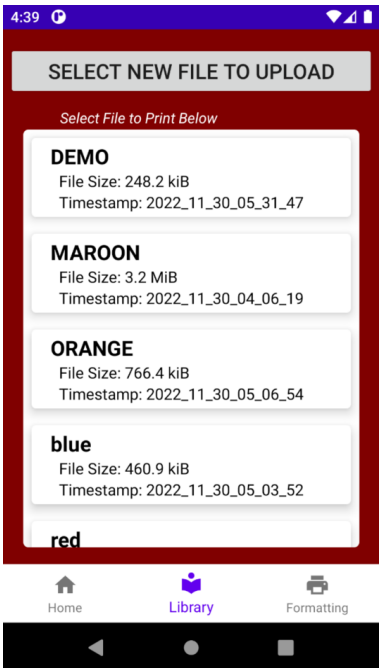


Figure 44. File Showing in Library

4.5.2. Home Tab and Firebase Realtime Database Validation

The following figures 45-47 show a working home tab in conjunction with a working Firebase realtime database. The figures demonstrate values in FRD being updated when the user has selected a file and chooses to push the Start Print button and the Cancel Print button.

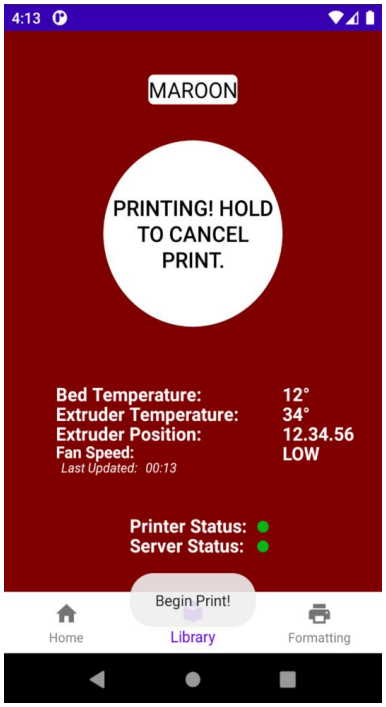


Figure 45. Home Tab Starting Print

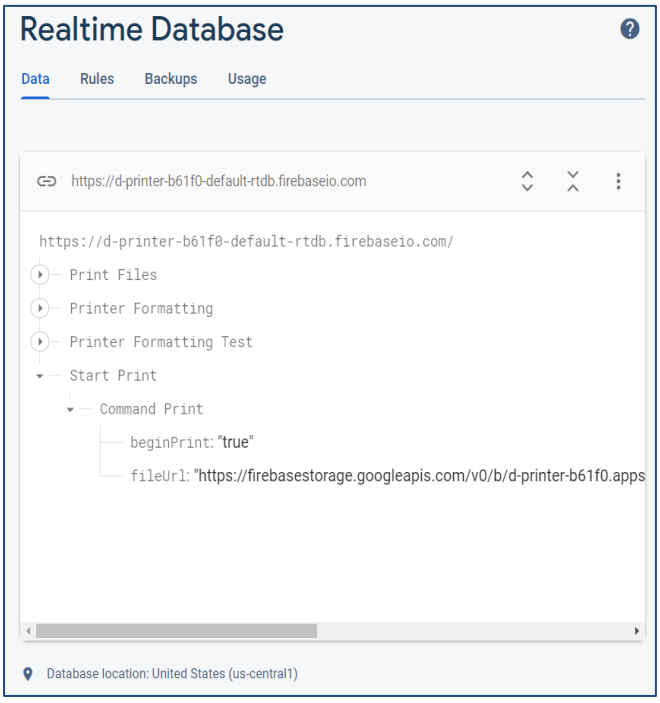


Figure 46. Firebase Realtime Database Starting Print Update

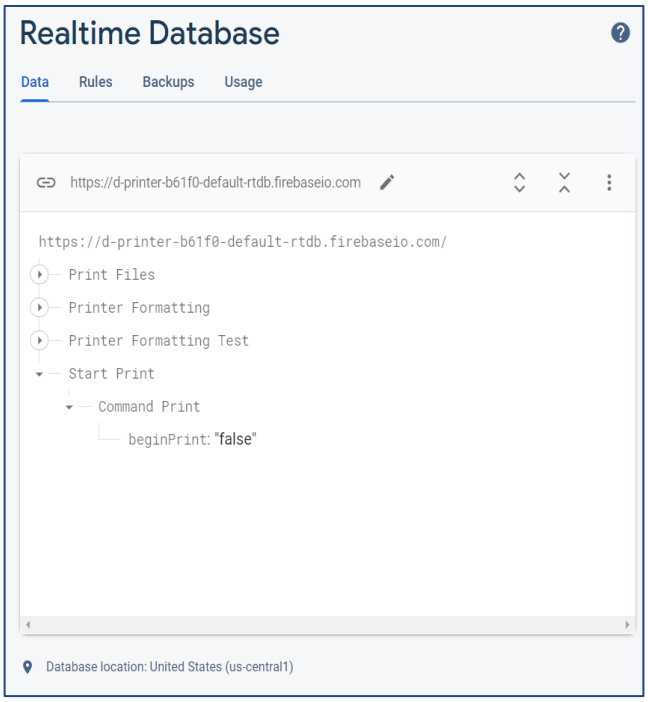


Figure 47. Firebase Realtime Database Stopping Print Update

4.5.3. Formatting Tab, Home Tab, and Database Validation

The following figures 48-50 show a working formatting tab in conjunction with FRD and the home tab. The following figures demonstrate the FRD being updated when a user inputs and applies new formatting values, and the home tab displaying newly changed formatting values which in the future will display formatting values provided by the MCU.

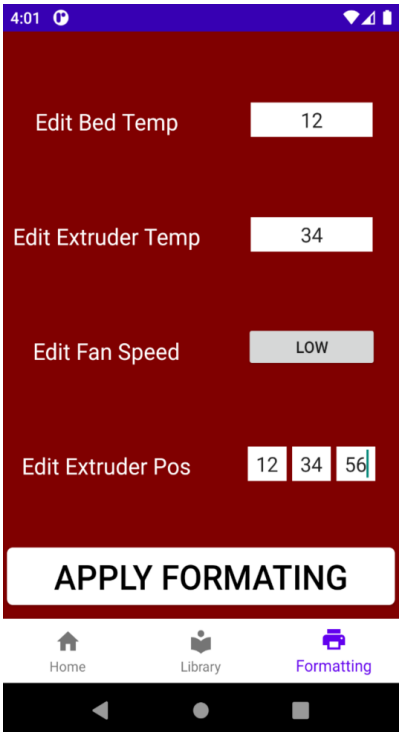


Figure 48. Formatting Tab User Input Values

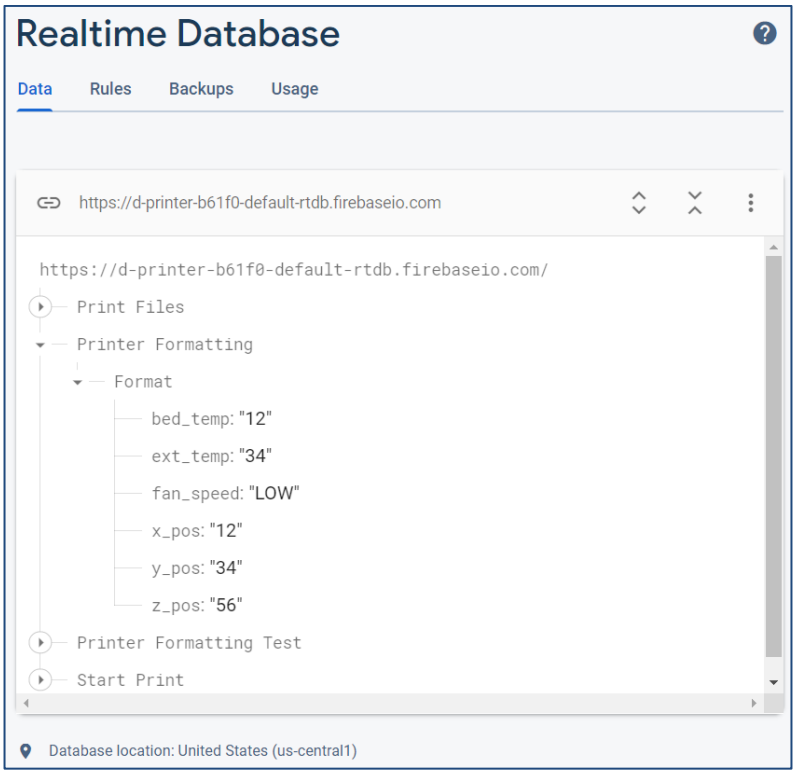


Figure 49. User Input Values Stored in FRD

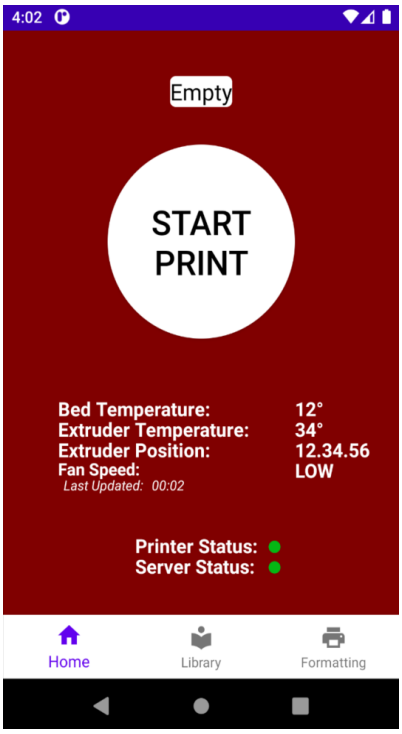


Figure 50. Home Tab Formatting Display

4.6. Application Bugs

4.6.1. Navigation Bar

The navigation bar fails to follow the main display view. When users select a file to print, the main display shows an updated home tab with the file name at the top. However, the Navigation bar at the bottom still shows the library tab.

4.6.2. URL to FRD

Over several weeks, the communication of the URL to the FRD has been evaluated and works a majority of the time. When evaluated over twenty times, the URL will likely fail only once. FRD does not handle large data well which is why the FS is incorporated into the library. The URL string may be too long to store and therefore fails to upload along with the rest of the file information such as name, size, and timestamp.

To reduce the chances of URL upload failure, multiple redundancies were added such as performing the same line of code to upload the file more than twice if the URL was not uploaded to FS.

4.7. Phone Application and Web-Based Server Conclusion

The two bugs mentioned above are currently being worked on with solutions such as using live data for controlling the navigation bar directly and other ways to upload to Firebase Realtime Database. Overall, the application and server subsystem are working well. All

features of the original 3D printer have been implemented within the application, and except for the URL bug, communication of all features between the application and server has been successful. During the break between semesters, solutions for both bugs will be heavily focused along with any adjustments needed to the current subsystem to smoothly incorporate the MCU subsystem.

3D Printer and Application Interface

Cody Hutchison
Steven Liu
Abigail Morar

System Description and Development

REVISION – 0.0
30 April 2023

System Description and Development
FOR
3D Printer and Application Interface

Team 21

Approved by:

_____
Cody Hutchison Date

_____
Prof. Lusher Date

_____
Dalton Cyr Date

Change Record

Rev.	Date	Originator	Approvals	Description
0.0	4/30/2023	Dalton Cyr	Cody Hutchison	Draft Release

Table of Contents

List of Figures

List of Tables

1. Overview

This 3D printer system aims at providing users remote control of a physical 3D printer, freeing users of manual or physical interaction with the printer to start or cancel a print. The system consists of three main subsystems: printed controller boards, microcontroller unit, and mobile application which included a smaller server subsystem to handle remote communications. Development of this project lasted two semesters with the first involving research, planning, and execution with the goal of finishing each subsystem. The following semester focused on execution, revision, and validation with the main goal of an integrated 3D Printing system. Currently, the 3D printing system is not fully functional due to some unfortunate issues within the MCU which is necessary for integrating all subsystems together.

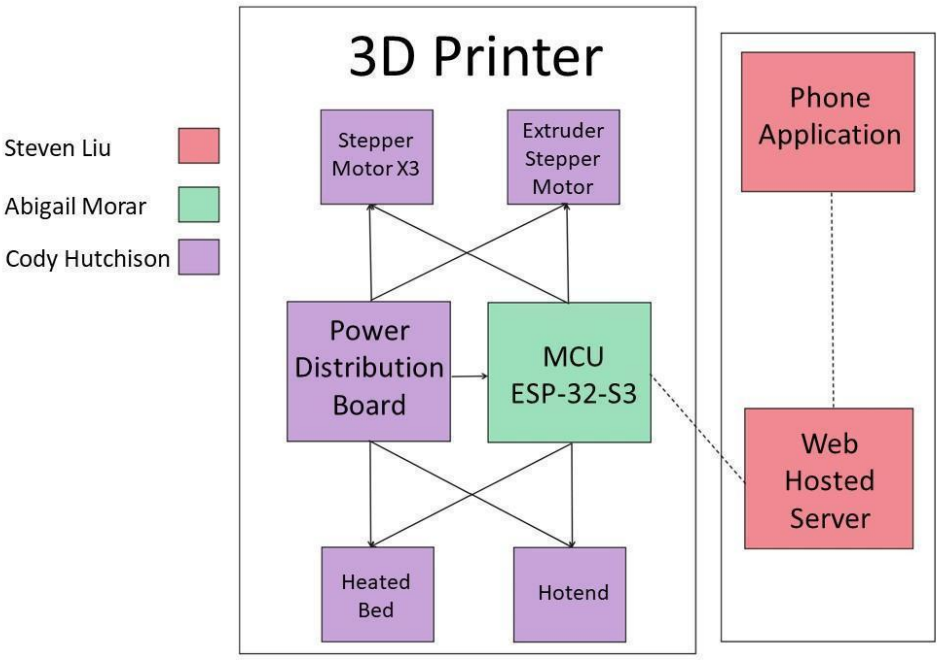


Figure 1. 3D Printer Block Diagram

2. Development Plan and Execution

The following sections below provide details on our design plan, execution, and validation of our 3D Printer and Application Interface project.

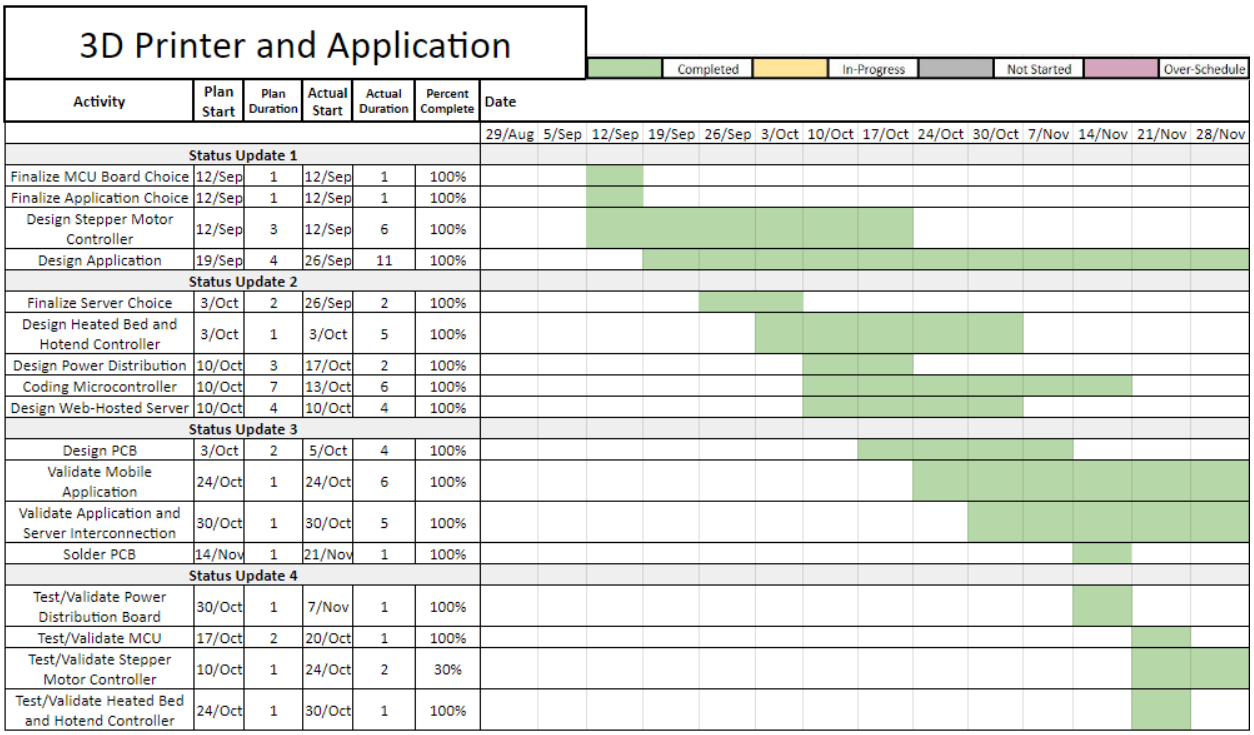
2.1. *Design Plan*

Upon introduction of the project, it was determined that there would be three main subsystems: printed controller board, MCU, and application. To further develop our design plan for this project, we first researched our individual subsystems and how well they would work together. It was decided that an MCU with Wi-Fi capabilities and a web-hosted server would be necessary to enable remote communication between the application and MCU subsystems. The server would provide print and formatting commands to the MCU, communicate updates from the MCU, and store G-code print files.

In the first semester of this project, we made foundational project decisions such as choosing board requirements for the PCBs, ESP32-S3 for the MCU, Android package (APK) file for the application, and Firebase for the server. For each of the subsystems, there were several design requirements that we established and aimed to achieve in order to have a fully operational 3D printer system which can be seen in the validation plan further below and in the FSR portion of the report. The goal by the end of the first semester was to have finished and functional individual subsystems. The second semester focused on completing those subsystems, integrating them, and performing revisions to the subsystems or system as a whole if issues arose. The goal for the end of the second semester was to have a validated and fully integrated system. To accomplish these goals, we set smaller individual goals throughout both semesters in an execution chart which can be seen below and is elaborated on in the execution portion of this document.

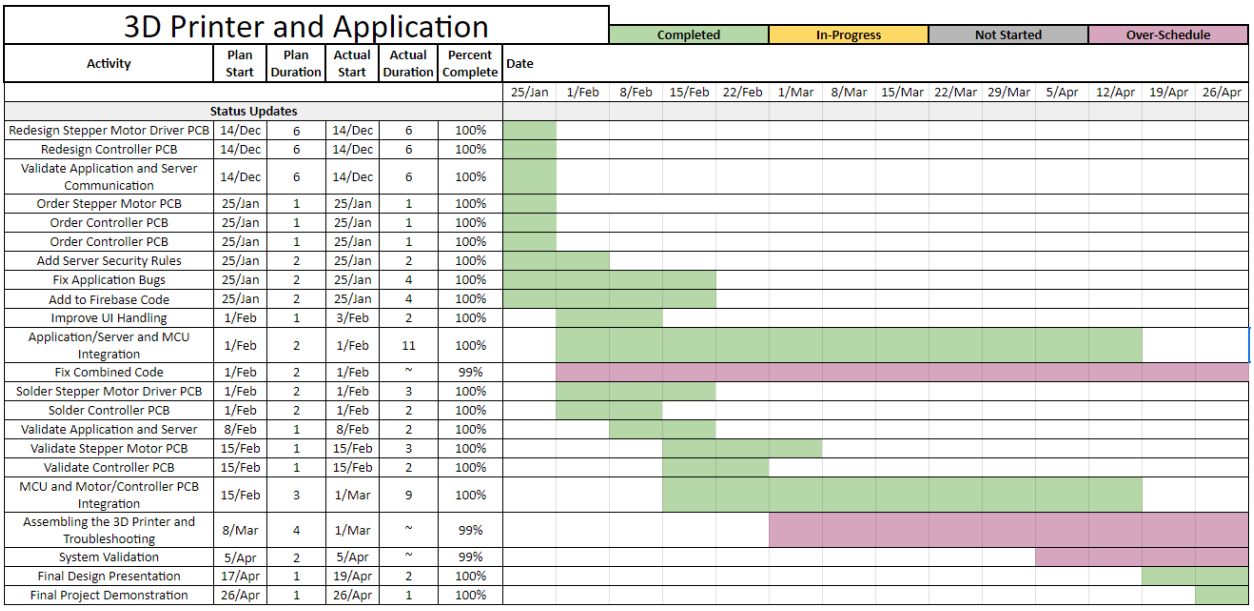
2.2. *Execution*

Keeping our functional system requirements in mind, we developed an execution plan for each semester that details our goals on a weekly basis. In the execution charts below, you can also view our progress on each of these tasks. During the first semester, the primary goal was the designing and assembling of the PCB and the designing and programming of the MCU, application, and server.



Activity	Plan Start	Plan Duration	Actual Start	Actual Duration	Percent Complete
Status Update 1					
Finalize MCU Board Choice	12/Sep	1	12/Sep	1	100%
Finalize Application Choice	12/Sep	1	12/Sep	1	100%
Design Stepper Motor Controller	12/Sep	3	12/Sep	6	100%
Design Application	19/Sep	4	26/Sep	11	100%
Status Update 2					
Finalize Server Choice	3/Oct	2	26/Sep	2	100%
Design Heated Bed and Hotend Controller	3/Oct	1	3/Oct	5	100%
Design Power Distribution	10/Oct	3	17/Oct	2	100%
Coding Microcontroller	10/Oct	7	13/Oct	6	100%
Design Web-Hosted Server	10/Oct	4	10/Oct	4	100%
Status Update 3					
Design PCB	3/Oct	2	5/Oct	4	100%
Validate Mobile Application	24/Oct	1	24/Oct	6	100%
Validate Application and Server Interconnection	30/Oct	1	30/Oct	5	100%
Solder PCB	14/Nov	1	21/Nov	1	100%
Status Update 4					
Test/Validate Power Distribution Board	30/Oct	1	7/Nov	1	100%
Test/Validate MCU	17/Oct	2	20/Oct	1	100%
Test/Validate Stepper Motor Controller	10/Oct	1	24/Oct	2	30%
Test/Validate Heated Bed and Hotend Controller	24/Oct	1	30/Oct	1	100%

Figure 2. Fall 2022: Execution Plan



Activity	Plan Start	Plan Duration	Actual Start	Actual Duration	Percent Complete
Status Updates					
Redesign Stepper Motor Driver PCB	14/Dec	6	14/Dec	6	100%
Redesign Controller PCB	14/Dec	6	14/Dec	6	100%
Validate Application and Server Communication	14/Dec	6	14/Dec	6	100%
Order Stepper Motor PCB	25/Jan	1	25/Jan	1	100%
Order Controller PCB	25/Jan	1	25/Jan	1	100%
Order Controller PCB	25/Jan	1	25/Jan	1	100%
Add Server Security Rules	25/Jan	2	25/Jan	2	100%
Fix Application Bugs	25/Jan	2	25/Jan	4	100%
Add to Firebase Code	25/Jan	2	25/Jan	4	100%
Improve UI Handling	1/Feb	1	3/Feb	2	100%
Application/Server and MCU Integration	1/Feb	2	1/Feb	11	100%
Fix Combined Code	1/Feb	2	1/Feb	~	99%
Solder Stepper Motor Driver PCB	1/Feb	2	1/Feb	3	100%
Solder Controller PCB	1/Feb	2	1/Feb	2	100%
Validate Application and Server	8/Feb	1	8/Feb	2	100%
Validate Stepper Motor PCB	15/Feb	1	15/Feb	3	100%
Validate Controller PCB	15/Feb	1	15/Feb	2	100%
MCU and Motor/Controller PCB Integration	15/Feb	3	1/Mar	9	100%
Assembling the 3D Printer and Troubleshooting	8/Mar	4	1/Mar	~	99%
System Validation	5/Apr	2	5/Apr	~	99%
Final Design Presentation	17/Apr	1	19/Apr	2	100%
Final Project Demonstration	26/Apr	1	26/Apr	1	100%

Figure 3. Spring 2023: Execution Plan

2.3. *Validation Plan*

We are able to control all aspects and components of the 3D printer. Temperatures of the hot head and heated bed are set and maintained through the application when chosen by the user or executing a G-code file. The fans operate and update accordingly with the same input methods. The axis motors move smoothly, quietly, and accurately to the user specified input. The motors allow the extruder to reach the full print area of 220mm x 220mm x

250mm as originally designed and the limit switches perform as intended when the G28 (auto-homing) command is called. We are unable to validate the system fully as there is a bug when executing G0 and G1 commands. The fans, heated components, and all other G-code commands operate as programmed; however, the motors execute a previous move command when the subsequent values being parsed remain unchanged.

FSR ref.	Validation	Success Criteria	Test Bench	Status	Responsible
3.2.1.2.	Stepper Motor Circuit	Output of 5V 0.84A on three specific outputs, 5V 1A on another output with specific controller inputs	Use Function Generator, DC power generator, controller board, and multimeter to validate output requirements.	Passed	Cody Hutchison
3.2.1.3.	Heated Bed Circuit	Output of 24V 9.167A when controller input is made	Use Function Generator, DC power generator, controller board, and multimeter to validate output requirements.	Passed	Cody Hutchison
3.2.1.3.	Hotend Circuit	Output of 24V 1.67A when controller input is made	Use Function Generator, DC power generator, controller board, and multimeter to validate output requirements.	Passed	Cody Hutchison
3.2.1.3.	Fan 1 and 2 Circuit	Output of 24V 100mA when controller input is made	Use Function Generator, DC power generator, controller board, and multimeter to validate output requirements.	Passed	Cody Hutchison
3.2.4.1.	Power Distribution Circuit	Ensure DC-DC convertor takes 24V input and outputs 5V	Use multimeter to validate all output possiblilites and connections.	Passed	Cody Hutchison
3.2.3.1.	Full Printer Control Features	User choosen printer formatting, file selection, and print initiation	Input printer formatting, select print file, select print initiation and check server for verification	Passed	Steven Liu
3.2.3.2.	Application File Chooser	Application succussfully selects device files	Perform file upload to server and check firebase storage for verification	Passed	Steven Liu
3.2.3.3.	Interactive Library View	Library displays print files that users can select	Attempt print initiation and check server and home tab for verfication	Passed	Steven Liu
3.2.3.4.	Server Storage	Stores print files	Upload print file and retrieve download url	Passed	Steven Liu
3.2.3.5.	Server Realtime Database	Saves print data and accessible print data	attempt printer formatting change and file upload, verify data in server and library tab	Passed	Steven Liu
3.2.3.6.	Application and Server Connection	Successful communication between application and server	Attempt all application printer features and verify communication to server	Passed	Steven Liu
3.2.4.2.	Communication of ESP32	Check incoming transmission from server	Connect MCU to network and send pings to server.	Passed	Abigail Morar
3.2.1.4.	Extruder	Ensure printer is extruding and retracting the proper amount of filament	Feed filament through opening and send code to force filament through nozzle by specified amounts.	Failed	Abigail Morar
3.2.1.	Microcontroller	Can communicate correctly with motors and heated components	Sending inputs through every channel to show everything is connected and operating correctly.	Passed	Abigail Morar
3.2.1.1.	Stepper Motors	Function smoothly and rotate accordingly by required distance	Various input cases will be used to track movement of the motors. This will be tested to ensure correct communication between board and device.	Passed	All
3.2.1.3.	Heated Bed Temperature	Can reach 220°C	Use infrared thermometer to check surface temperature.	Passed	All
3.2.1.3.	Hotend Temperature	Can reach 80°C	Use infrared thermometer to check surface temperature.	Passed	All
3.2.2.1.	Extruder Location	Can reach the full printing area of 220x220x250 (mm)	Set extrusion nozzle head to every possible coordinate.	Passed	All

Figure 4. Validation Plan

3. Critical System Data

As previously mentioned, this project consists of three subsystems. While the first semester concentrated on developing the individual subsystems, the focus of the second semester was on the integration of these subsystems and total system validation.

3.1. *Printed Control Board Data*

The primary use for the main controller PCB is to provide power and send/receive signals for the 3D printer. The principal power supply that converts the 110V AC into 24V DC provides up to 360W of power for the printer. The main controller PCB takes that 24V and splits it between the 24V systems and the 24V to 5V converter which relays it throughout the rest of the system. The stepper motor PCB holds the TMC-2130 stepper motor controllers that allow the stepper motors to function correctly with inputs from the MCU and 5V from the converter. The total power used by the system is only 245.4877W, which leaves over 100W of spare power if needed in future upgrades.

Device	Voltage (V)	Amperage (A)	Power (W)
Heated Bed	23.68	8.5	201.28
Hot Head	23.921	1.7	40.6657
Fan 1	24.028	0.068	1.6339
Fan 2	24.028	0.047	1.1293
X-Driver	2.456	0.224	0.5501
Y-Driver	2.5266	0.0225	0.0568
Z-Driver	2.5058	0.0303	0.0759
Extruder	2.4881	0.0386	0.096

Table 1. 3D Printer Power Usage

3.2. *MCU Data*

The MCU is the hub of the 3D printer. It reads in commands from the application through Firebase, sends signals through the PCBs to the printer components, and computes information back from the printer to the phone user. Reading and writing to Firebase is instantaneous. The speed of execution of the commands from the app are dependent on the strength of Wi-Fi connection. Generally, each command is carried out within a couple seconds from when the user updates Firebase. Each function of the MCU program is detailed below and the full program can be found in Team 21's GitHub repository.

3.2.1. setup()

Where the MCU gets connected to the Wi-Fi and Firebase server. All defined MCU pins are set to an input or output and are initialized. Additionally, all header files, pin definitions, Firebase and Wi-Fi credentials declared, Firebase paths, and global variables and constants are called before the setup function.

3.2.2. gcode()

The MCU gets the G-code's file name and total number of nodes for file print. Nodes containing G-code lines are obtained and parsed to get individual commands in order. Commands beginning with G's deal with linear movements, extrusion, and mode positioning. They are parsed further by their command numbers and other parameters dealing with stepper motor movement are tokenized, calculated, executed, and stored to be updated back to Firebase. Commands beginning with M's are also parsed further by their command numbers and subsequent parameters. These commands mostly deal with temperature settings and fan speeds. After values are calculated, compared, and set, they are stored to be updated back to Firebase. After each command, the newly stored values get reported to Firebase for realtime updates to the user. The server is then checked for updates from the user if they decided to change fan speeds, temperatures, or cancel the print early. After every node of the G-code file is executed, the MCU will check the temperatures of the heated bed and hot head again to make sure they are being properly maintained. This process is iterated for every line of G-code in every node of the file until the last command is finished. Once the 3D print is complete, the status of the print is sent back as "true", the last values of the printer are returned to Firebase, and the function has concluded.

3.2.3. parse()

Similar to the gcode() function, this one parses the incoming commands from the user input. For the case like G0, feedrates have a default speed in this section. The axis motors are still moveable; however, there's no extrusion capabilities for the user to manually adjust. The temperature and fan setting commands operate the same in this function as in the previous. Once one command is executed, the new value gets reported back to the user through Firebase.

3.2.4. firebase_report()

This function gets called from other functions often after performing commands. It sets all the updated values into strings to be written back to Firebase for the user to see realtime changes from the 3D printer.

3.2.5. firebase_check()

User input of new extruder location, temperature changes, and fan settings are checked constantly from the server to update the system. The user controlled settings are set to be in absolute positioning mode like default G-code settings. When a print job is active, the user can continue changing the heated bed, hot head, and fan speed; however, the user no longer has control over the location of the extruder until the printing status is "false" again. After each command is checked and found, the value gets sent to the parse() function. This functionality is the same as the original user interface.

3.2.6. loop()

The loop() function is the main function that continuously runs until the 3D printer gets shut off. It contains a delay to not overwhelm the MCU and checks the print status to see whether a G-code file should begin printing or not. If a print is selected to run, the gcode() function will be executed until it completes. If not, the program runs through the firebase_check(), parse(), and whatever other commands/functions that follow.

3.3. *Server Data*

The server acts as a bridge between the mobile application and MCU subsystems by providing storage for print files and print commands that can be read and written from both

sides. The Firebase Realtime Database is currently at a low 6.1% capacity and has a max of 1GB of storage, and to stay within the free tier of Firebase services only 360MB/day of downloads. Over the last few weeks of this project, we have exceeded the free tier limit on downloads by 427.9MB resulting in a cost of $1.85.

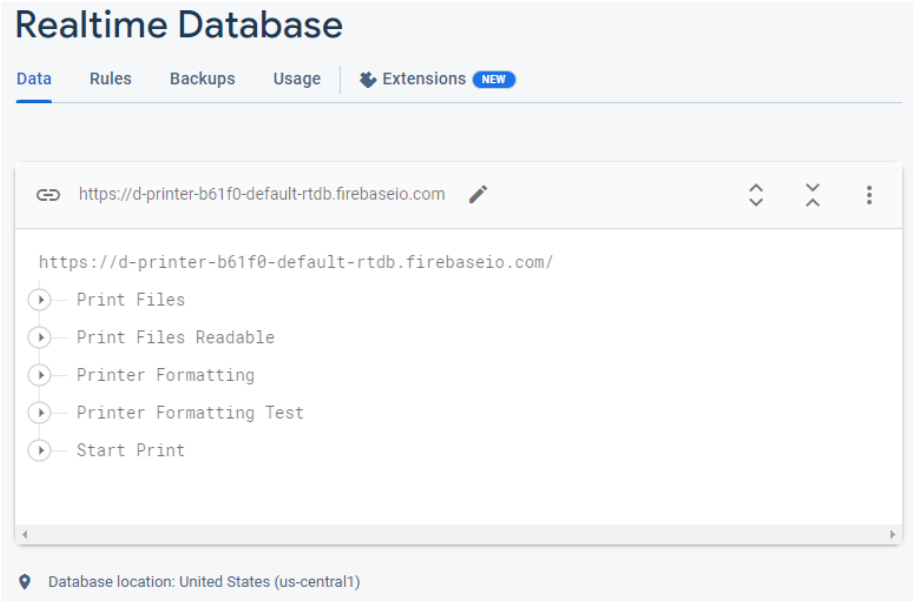


Figure 5. Firebase Realtime Database Overview

3.3.1. Print Files

The Print Files node holds the relevant file information such as file name, size, timestamp of upload, and the number of nodes the file occupies in the Print Files Readable node.

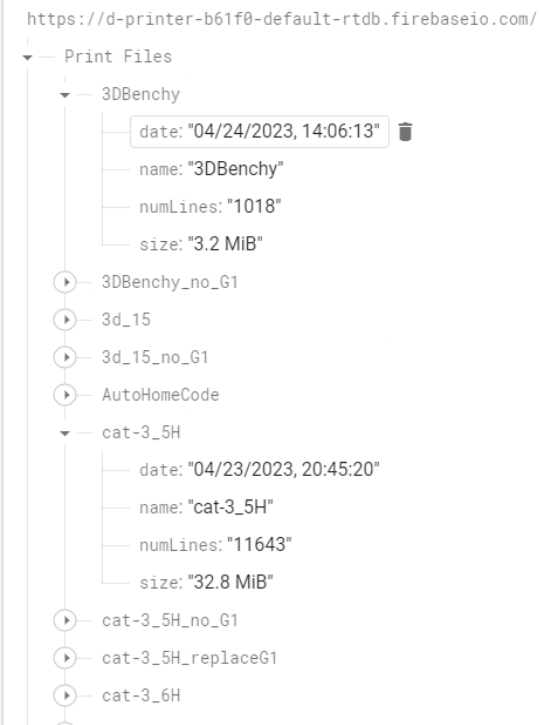


Figure 5. Print Files Node

3.3.2. Print Files Readable

Since we had to change the location of the files from Firebase Storage to Firebase Realtime Database, the Print Files Readable node now contains an endless supply of nodes for every saved G-code file. Each resulting node contains up to 100 parsed G-code lines for the whole file, if any are uploaded to the server.

Figure 7. Print Files Readable Node

3.3.3. Printer Formatting

The Printer Formatting node holds printer formatting values that the user chooses from the application, unless a print job is ongoing. The 3D printer's components get updated with this information once it is read by the MCU. After execution of the commands, the MCU then reports the newest physical changes of the 3D printer to the Printer Formatting Test node. This node is directly connected to the phone application's Home tab.

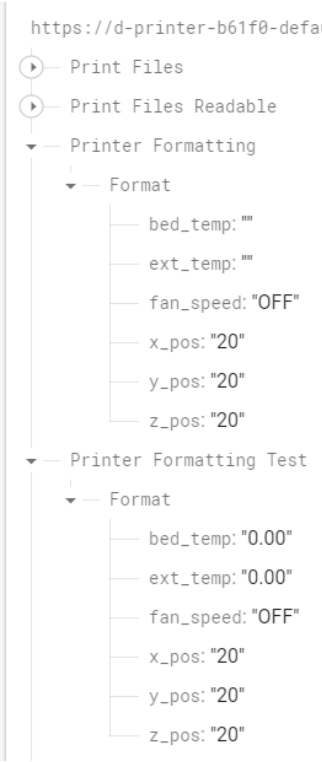


Figure 8. Print Formatting Node

3.3.4. Start Print

The Start Print node has been updated twice since the subsystem report. It now includes fileName, fileNumLines, and print_complete nodes along with the previous beginPrint node. These nodes provide the MCU with all the information it needs to find, read, parse, and execute a G-code file.

Figure 9. Start Print Node

3.4. *Application Data*

The application provides the user interface for the 3D printer system. It handles all user inputs, formats them, and communicates them with the server. The application has three different tabs: Home, Library, and Formatting. The final version of the application is shown below along with some information on communication speeds.

3.4.1. Home Tab

Below is the final version of the application. A few changes were made visually such as the removal of the status dot and color values for the background. After running numerous tests, the communication to and from the application and server seems instantaneous depending on the user's internet service.

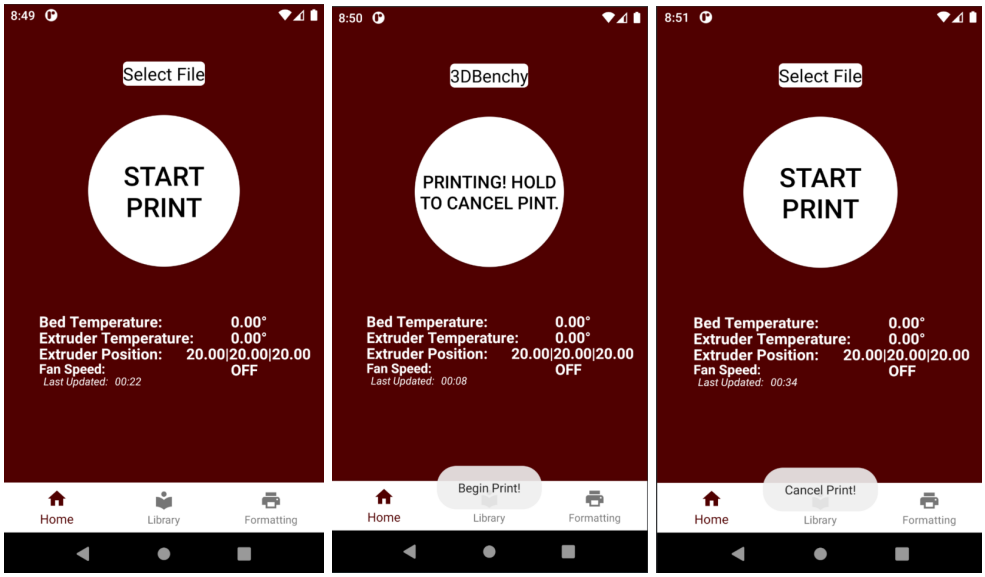


Figure 10. Home Tab

3.4.2. Library Tab

Below is the final version of the Library tab. The tab now has the responsibility of parsing G-code files and has updated programming to prevent the upload of non-G-code files and duplicate files already existing in the server. When adding files, the upload speed is approximately 40kB/s. Although the specific speed cannot be calculated at this time, the approximate speed was calculated through manual stopwatch and watching the file populate the database.

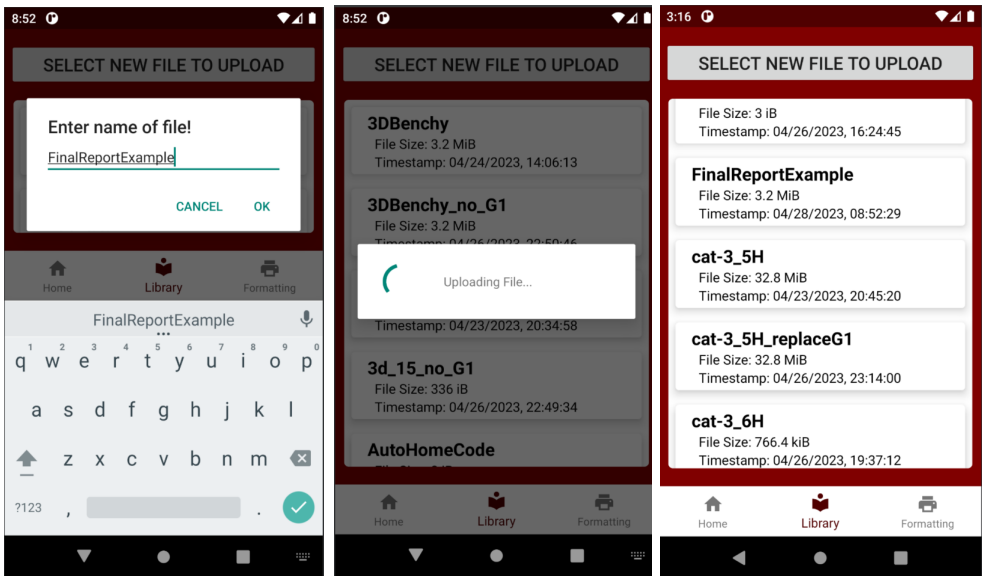


Figure 11. Library Tab

3.4.3. Formatting Tab

The Formatting tab below has not changed since the subsystem report.

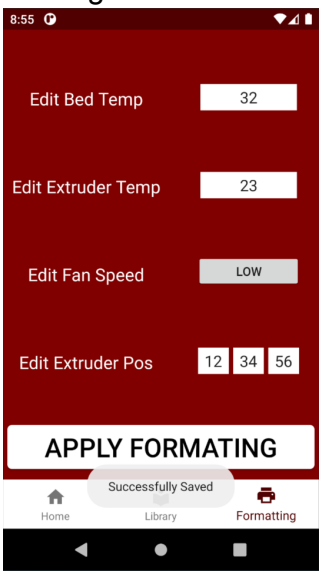


Figure 12. Formatting Tab

4. Conclusion

Despite our team's hard work and dedication, the 3D Printer and Application Interface system encountered a small, but critical bug that prevented us from fully reaching our intended goal. During the integrating process, our team made significant progress in each of our subsystems, which showed great potential for a complete working system. The almost fully functional system was not for lack of effort as we demonstrated persistence through several challenges which we overcame. Although we were ultimately unsuccessful in accomplishing every goal in our FSR report, this project served as a valuable learning experience for the team as it highlighted the importance of teamwork, perseverance, and adaptability when faced with challenges.

4.1. *Issues*

The biggest problem affecting the 3D printer is the bug encountered when running certain G-code files. This bug reveals different results depending on the commands it reads in. As mentioned earlier, if the location of the extruder is told to maintain its current position during a subsequent command, it will execute the previous move command again anyway. However, when a user tries to mimic these problem G-code lines by updating the Formatting tab, the extruder stays in its desired location as intended. Additionally, when a print job is being executed, the motors move as if they were set to relative positioning mode, meaning that the movement of the extruder is building off of the previous command rather than its location being absolute to the coordinates of the board. Evidently, this prevents a successful print since some move commands are iterating twice over, causing the extruder to try printing outside the bounds. The MCU program has been reviewed by several people outside of our team and all were unable to find where the specific issue lies within the code itself.

Other than the bug, there were a couple of controls that were supposed to be implemented in the application so that they would match the original Ender 3 interface's capabilities. These include a button for setting the home position and one to disable stepper motors to allow for physical manual movement when the printer is on.

4.2. *Decisions*

Since narrowing down the issue, we attempted many configurations to solve the problem: changing steps per revolution, changing steps per mm, hard coding values in move commands, rewriting calculations and functions, and commenting out possible obstructive code. However, we were unable to find the exact issue to fully resolve it. We decided to make short G-code files to show how the printer parses and performs the other working G-code commands correctly when a print job is active. The functionality of our system is seen by successfully executing commands through the user input controls from the application. As our main focus was set on fixing the bug, we ran out of time to implement the two additional button controls mentioned above. Although we were unable to display a final 3D printed product, with our work-arounds, we were able to show how each individual component of the printer is in fact fully functional and completely controllable.

4.3. *Understandings*

Capstone was a great learning experience on how to work as a team on a long-term project. Over the nine months, the team had to learn how to dissect a problem and

individually design subsystems for future integration. Each member needed comprehensive knowledge of their system to resolve any issues that arose during integration. While the project did not quite meet the original goal, with a few additional weeks we could have possibly sorted out the bug and made the adjustments needed for proper printing.

On the technical side, each team member had to spend countless hours learning something new in order to design and build their subsystem for the 3D printer. For the PCB design, there was much time spent on learning how to use Altium to design the circuit. Part of designing the PCB was also understanding how the components operated, their size, and how they are soldered onto the board. Most of the components utilized were Surface Mounted Devices (SMD), and it was discovered that selecting the appropriate parts was important after encountering a capacitor that was smaller than the tip of a pencil. The other skill learned was soldering small components and SMD parts.

For the MCU programming, learning how G-code functions operate the printer components such as the stepper motors, heating elements, and fans is crucial to making a working subsystem. As the MCU is the base for full integration across all subsystems, learning how the pins of the MCU work and how it communicates with the electrical components and Firebase server is critical as well. During full integration, combining code and debugging was a tedious process that taught patience, resilience, and how to work through problems with teammates. Additionally, in order to find the necessary steps per revolution and steps per mm for each motor to function accurately requires understanding of the mechanical aspects of the stepper motors themselves.

For the application and server subsystem, the first month consisted of learning how to program using Kotlin inside the Android Studio IDE. To complete this subsystem, It was necessary to understand how Kotlin, the backend code, and the XML user interface interacted with each other, as well as how to perform internal and external communication (e.g. between tabs or with the Firebase server). Valuable skills learned were how to build a mobile application, set up a server, and project planning for integration not only between the application and server but also with the MCU.

4.4. *Future Work*

In addition to what we did not get to resolve, there are a few items that the team thought would be smart to add, update, or finish due to time constraints in the project:

- Redesign of the main controller PCB to correctly fix the thermistor and switch circuits
- Combine both PCBs into one board and change placement of the board inputs
- Add the ability to change the home location of the extruder
- Add a variable control to the fans instead of just on/off
- Add the ability to have multiple users
- Add connection status icons for the user
- Track filament usage
- Always know location of extruder whether manually moved or moved after power off
- Remote power on/off feature
- Set up Wi-Fi through phone application