Mobile 3D Printer: EECS Project Document

Sean Booth, Samuel Lewis, Preston Pickering, and Garrett Hallquist

6 May 2022

Contents

1 Overview

1.1 Executive Summary

The Mobile 3D printer team is working alongside a mechanical engineering team to produce a 3D printer which can print in an effectively unlimited area. It will achieve this by allowing the user to move the printer around to work on different sections of the model being printed. The printer will be able to print in five axis (X, Y, Z, and rotate around two of these), allowing it to also print a wider variety of models. It will utilize this additional capacity to print directly onto arbitrary surfaces other than a flat plane. Our goals for the project are for it to be able to print on two test surfaces: a quarter of a cylinder as well as on a ridge formed from the intersection of two sloped planes. If we can print on these surfaces, it will show that the printer can utilize its rotational axis as well as print models consisting of multiple parts.

The ultimate goal for these developments is to produce a printer which can work on part of a print job, complete that section, and then be moved by the user to another location in order to begin work on the next section of the overall model. Because of this intent, the printer can not be bound to a limited gantry system but should instead have a practically unlimited print area once mobility is taken into account. Along the way, other technologies might be utilized in order to achieve our goals. Standing out among these are the potential use of computer vision in order to allow the printer to orient itself in space after movement, as well as the use of a simulation in order to help design the printer and the algorithms which will be used to drive it. This is a large project, but we look forward to improving on existing 3D printing technology to produce a printer with extended capabilities.

The team is currently in the early design phases of the project. The mechanical team has just selected a general design, and we are ramping up to begin designing our portion of the system in earnest over the coming weeks.

1.2 Team Communication Protocols and Standards

Contact Info

- **Sean Booth**, ECE, Email/Microsoft Teams: boothse@oregonstate.edu
- **Samuel Lewis**, CS, Email/Microsoft Teams: lewiss4@oregonstate.edu
- **Preston Pickering**, CS, Email/Microsoft Teams: pickerip@oregonstate.edu
- **Garrett Hallquist**, ECE, Email/Microsoft Teams: hallquig@oregonstate.edu

Topic	Protocol	Standard
Team Meetings	A number of weekly meetings will occur. A Monday meeting to plan out the next week, a Wednesday night meeting to touch base and record the progress video, and a Thursday meeting with the project partner and ME team.	Team members will attend meetings. If they cannot attend meetings, they will inform others that they cannot make it in advance. If team members have to miss meetings, the other members will post a meeting summary to discord.
Task Management	Regular meetings will be used to handle task management and accountability	At the weekly Monday meeting CS and ECE will go over progress and update tasks to complete. Anything which members know they want to talk about ahead of time should be put on the agenda document for that week
Deliverables	All deliverables will be reviewed by other team members before submitting.	Once work has been reviewed and corrected it can be submitted onto Canvas for grading.
Project File Storage	All files specific to the EECS team will be stored in the shared Google Drive. Anything which needs to be shared with the ME team will be put in their Box drive. Any code which multiple people will be collaborating on will be stored in a private Git repository.	Team members will store files in the listed locations.

Table 1: Protocols and Standard Table

Project Partner Communication Preferences

- All groups will communicate mainly through Microsoft Teams. Groups will have channels to communicate amongst themselves but are strongly encouraged to talk in general channels.
- There is a weekly design meeting held on Thursdays in Dearborn Hall with the project partner, their three mechanical engineering graduate students assigned to the project, and the mechanical engineering capstone team.

1.3 Gap Analysis

Current 3D printing solutions for large scale 3D printers are fairly limited to using fixed stands, gantry systems, or moving the structure that is being printed on. Most current mobile 3D printers are limited to concrete printing and construction [1]. These printers take up huge amounts of space, and are often less precise. This project aims to create a prototype 3D printer to address these gaps in 3D printer technology. The prototype 3D printer is compact, movable, precise, and capable if printing on a stationary non-planar surface. The prototype printer will be printing plastic, but is intended to act as a model for 3D printing metal. This could be used for surface finishing, printing on curved surfaces, and printing larger structures without also needing a larger printing bed. The greatest benefit if the mobile 3D printer is it's incredible versatility. Most industrial 3D printers are highly specialized, meaning they can do a few operations really well, but cannot be easily re-purposed for an entirely different job. The mobile 3D printer has far less space and functional limitations, so it can easily be used for many different jobs without having to change the printer itself. This technology could be utilized by manufacturing companies such as Boeing and OMIC R&D who would benefit from a more versatile 3D printer configuration.

1.4 Proposed Timeline and Milestones

	PROJECT TITLE	Mobile 3D Printer	Fall Term									
	PROJECT PARTNER	Matthew Campbell										
WBS NUMBER	TASK TITLE	TASK OWNER	% OF TASK COMPLETE	WEEK 1	WEEK 2	WEEK 3	WEEK 4	WEEK 5	WEEK 6	WEEK 7	WEEK 8	WEEK 9	WEEK 10
PHASE 1: Research and Design													
1.1	Meet with Partner and Discuss Scope	All	80%										
1.2	Preliminary Research	All	55%										
1.3	Teamwork/Weekly Meetings	All	Continuous				Project Partner U			Project Partner U	Teamwork Reflec		Project Partner U
1.4	Design System	All	0%										
1.5	Project Document	All	33%				Section 1 Draft D	Section 2 Draft D	Section 1/2 Peer I	Section 1 and 2 D	Section 3 Draft D	Section 3 Peer Re	**Sections 1-3 Due**
1.6	Design Impact Assessment	All	70%					Draft Due	Peer Review Due				**Final Draft Due**
1.7	System Block Diagram	All	0%										
Phase 2: Prototyping													
2.1	Tech Demo: IK Simulation	Preston P	5%										
2.2	Tech Demo: Microcontroller PCB Design	Garrett H	6%										
2.3	Tech Demo: Power Supply	Sean B	5%										
2.4	Tech Demo: Vision System	Sam L	5%										

Key	
Red	Critical Path
Yellow	Active Work

Figure 1: Fall Term Gantt Chart

1.5 References and File Links

1.5.1 References

[1] D. Sher. "The top 10 biggest 3d printers." (2015), [Online]. Available: https://3dprintingindustry.com/news/top-10-largest-3d-printers-54377/.

1.5.2 File Links

1.6 Revision Table

Date	Revision Info
10/22/2021	Initial document revision.
10/22/2021	Preston Pickering: Added executive summary.
10/22/2021	Sean Booth: Set up document format and bibliography. Revised content.
10/22/2021	Garrett Hallquist: Added Gap Analysis.
10/29/2021	Garrett Hallquist: Gap Analysis - Added Davide Sher source.
10/29/2021	Samuel Lewis: Team Communication Protocols and Standards - Set up table and filled out Team Meetings, Task Management, and Deliverables.
10/29/2021	Sean Booth: Updated bibliography file.
10/29/2021	Preston Pickering: Added figure labels as requested by Rachel Kate. Revised executive summary and made project partner communication more concrete based on feedback from the same.
11/12/2021	Garrett Hallquist: Gap Analysis - Added some content on printer versatility based on feedback.
11/12/2021	Preston Pickering: Executive summary - Clarified nature of printer mobility, elaborated on the need for the test surfaces, and improved professional voice. All suggestions from peer reviews.
11/12/2021	Samuel Lewis: Team Communication Protocols and Standards - Fixed error in table and revised grammatical errors in the section as a whole.
11/12/2021	Preston Pickering: Team Communication Protocols and Standards - Added a couple of standards based on peer review, added disciplines to team contact info list.
11/12/2021	Garrett Hallquist and Preston Pickering: Timeline - Created Gantt chart.
11/19/2021	Sean Booth: Updated gantt chart page to be both properly rotated and in larger font for readability, as per feedback.
12/3/2021	Garrett Hallquist: added minor details to Contact Info and Project Partner Communication Preferences.

2 Requirements, Impacts, and Risks

2.1 Requirements

These requirements outline what the system should be capable of. As it is a 3D printer, it must be able to move the print head to arbitrary positions with good accuracy, and because of the 5-axis implementation, it requires a certain configuration of joints to achieve such motion. The system also must obey certain safety requirements, such as avoiding self-collisions and dealing properly with unexpected conditions such as collisions or thermal runaway.

2.1.1 Positional Accuracy

The system will move any motor in the system accurately to within $\pm 1^\circ$ of the target position.

Verification process:

1. The system will be instructed to move to a well-defined position within its range of motion.
2. The system will then attempt to move to this position.
3. The difference between the actual position and the targeted position will be measured.

4. This process will be repeated at least 5 times.
5. If the system is always within the threshold, it passes this requirement.

2.1.2 Five Motor Control

The system will control 5 motors at once.

Verification process:

1. The system will be instructed to move all the motors at the same time.
2. All 5 of the motors must move during this test, approximately the same amount.
3. If the system can move all 5 motors simultaneously, it passes this requirement.

2.1.3 Filament Extrusion

The system will extrude filament to an accuracy of $\pm 5 \frac{\text{mm}}{\text{m}}$.

Verification process:

1. The system will be instructed to extrude 1 m of filament.
2. The system will then attempt to extrude this amount of filament.
3. The difference between the target and actual extrusion length will be measured.
4. This process will be repeated at least 5 times.
5. If the system is always within the threshold, it passes this requirement.

2.1.4 Software Self Collision Avoidance

The system's software will prevent motors from moving to angles which would cause a self-collision or otherwise be impossible due to the mechanical frame. This is not meant as a replacement to hardware collision avoidance, but as an extra layer of safety.
Verification process:

1. The system will be instructed to move in a way that would put a motor angle out of bounds.
2. The system's software will then abort the print job.

2.1.5 Arbitrary Positioning

The system will rotate motors independently to at least 20 degrees from its starting position

Verification process:

1. The system will be set to a known position.
2. The system will be instructed to move to a point where all motors are more than 20 degrees from their original position
3. The system will then attempt to reach this point.
4. If the system is able to move the motors, it passes the requirement.

2.1.6 Collision Detection

The system will detect if it has collided with an object or itself and stop all motors.

Verification process:

1. The system will be instructed to move to some position.
2. The system will then be physically obstructed so that it collides with a safe object.
3. The system should then stop all movement when it detects this collision.
4. This process will be repeated at least 5 times.
5. If the system stops every time it collides with an object, it passes this requirement.

2.1.7 Instruction Interface

The system will accept instructions through a wired serial interface, and will acknowledge the receipt of instructions. Invalid instructions are to be given an error message in reply.

Verification process:

1. The system will be given some valid or invalid instruction.
2. The system will then acknowledge the instruction, and give an error if it is invalid.
3. If the system responds correctly to the instruction, it passes this requirement.

2.1.8 Thermal Runaway Protection

The system will attempt to detect and stop any instances of thermal runaway. If expected and real temperature do not match for 10 s, the system will shut off all heating elements and require user intervention before it can start again.

Verification process:

1. The temperature sensor will be removed from the influence of the heating element.
2. The system will be instructed to heat up the hotend to 200 °C
3. The system should shut off the heating element within 20 s when it detects the fault.
4. If the system shuts down the heating element properly, it passes this requirement.

2.1.9 Safety

The system will contain a physical shutoff switch that can be used to disconnect power from the system in case of unexpected behavior. The system will also enclose any high voltage electronics and warning labels will be placed where hazards (such as hot areas) are present.

Verification process:

1. The system will be powered on.
2. The shutoff switch will then be pressed, and the system should immediately lose all power.
3. The system will also be visually inspected for exposed high voltages, and unlabeled hazards.
4. If the system shuts off as expected and there are no unlabeled, exposed hazards, the system passes this requirement.

2.2 Design Impact Statement

There are a number of ways this project, if brought to a larger scale, could have both positive and negative potential impacts, which will be discussed here in brief. Mitigation strategies for potential negative impacts will also be given. Impacts in the following areas will be touched upon: public health, safety, and welfare impacts; social and cultural impacts; environmental impacts; and economic factors.

2.2.1 Public Health, Safety, and Welfare Impacts

The two most pertinent potential negative health and safety impacts that this technology poses are not unique to this particular design, but are impacts derived from 3D printing and manufacturing as a whole, as well as from human-machine interactions. 3D printers present a risk of respiratory illness and pollution due to the materials commonly used as filaments. Melting these various plastics to be able to extrude them and shape them into useful parts and structures causes the release of harmful fumes rich in Volatile Organic Compounds [1]. ABS (Acrylonitrile Butadiene Styrene) is particularly notorious for this, but it also has the upside of being stronger than filaments with lower melting points such as PLA (Polylactic Acid). A preliminary study revealed a high presence (59%) of respiratory symptoms among a group of 46 workers, all of whom regularly work with 3D printers, as well as other, less common symptoms such as frequent headaches and cutaneous symptoms [2]. Luckily these health and emissions risks can be mitigated through the use of proper ventilation, filtration, and PPE.

The other pertinent health and safety risk is unintentional human-robot contact. At the current scale of the project, only two dangers are exposed to the human operators: that being pinch points, places where fingers or other extremities could become stuck while the printer is moving, and the hotend, which could cause burns if touched while in operation. If the project or a similar design were to be scaled up, there would be even more risk of damage from being hit by the moving parts of the device. A good mitigation strategy for impact-related injuries is to cover the larger parts of the device in foam, which can dampen the force of impacts with humans [3]. Both engineering and operational safeguards, as well as labeling of hazards, can also reduce the chance of an incident occurring, especially for the less obvious hazards.

2.2.2 Cultural and Social Impacts

The potential negative cultural and social impacts are once again derived from 3D printing technology as a whole, with the main concerns being that the new technology could contribute to socio-economic instability in developing nations and reduction of available jobs. When first deployed, the device will likely be very expensive and thus if it provides enough of an advantage, would contribute to socio-economic instability in developing nations, a concern expressed about 3D printing technology as a whole [4]. An open-source approach could limit the mitigate the problems that this might introduce, and even perhaps empower the formation of new industries around these advanced manufacturing technologies. Another concern, related to this first one, is the reduction of available manufacturing jobs due to this device, especially at larger scale, being able to replace a fair number of traditional manufacturing technologies. The mitigation stragety for this impact is education, creating higher-paying jobs working on maintaining and designing for these technologies [5].

2.2.3 Environmental Impacts

As the device is simply a 3D printer capable of printing on more surfaces, the environmental impacts are largely the same as well. The major downside of additive manufacturing compared to more traditional subtractive manufacturing, is the increased power usage from having to heat the filament, which makes 3D printers less efficient than something like a traditional CNC mill [6]. However, as mentioned in the previous source, 3D printers do perform better in minimal and maximal utilization scenarios, so the technology simply must be used responsibly. A potential benefit from this technology is enabling more parts to be produced in one location, reducing the need to ship parts from long distances away, thus reducing pollution from transportation [7].

2.2.4 Economic Factors

Potential negative economic impacts have been discussed in the section about cultural and social impacts, and the disruption to existing manufacturing jobs is likely to be the most pertinent impact. The increased localized production of parts that this technology enables could also lower global trade, as has been a concern with 3D printing as a whole. However, according to a research report from World Bank Group, 3D printing technologies may actually lead to an overall increase in global trade, despite some reductions from localization of production [8].

2.2.5 Conclusion

While there are definitely some health and safety concerns, as well as socio-economic risks associated with this technology, these potential risks can be mitigated via the strategies discussed in previous sections. There are also potential positive impacts that may empower developing nations to build innovative industry, and reduce transportation-based pollution.

2.3 Risks

Risk ID	Risk Description	Risk Category	Risk Probability	Risk Impact	Performance Indicator	Responsible Party	Action Plan
R1	Vendor Delay	Schedule	50%	M	Check intended shipment dates vs. actual shipment dates for all parts	Sean Booth	Reduce. Search for alternative vendors. If none, search for alternative parts
R2	Miscommunication With Mechanical Team	Organizational	80%	H	Stay in constant contact with mechanical team regarding project progress	Garrett Hallquist	Reduce. Schedule an emergency meeting with the mechanical team to discuss possible fixes or middle ground
R3	COVID-19 Resurgence	Public Health	20%	M	Monitor Oregon/OSU Health and Safety recommendations	Preston Pickering	Reduce. Reassign work that can be done remotely and set up frequent remote meetings
R4	Project Partner Changes Scope	Organizational	70%	L	Make sure project requirements are well communicated between all parties	Sam Lewis	Reduce. Remind the Project Partner of the Initially agreed upon requirements, and discuss current constraints
R5	Part Failure	Engineering	30%	L	Monitoring the printer during operation	Garrett Hallquist & Mechanical Team Rep	Retain. Immediately remove part before it can damage the rest of the system, then order/construct a replacement immediately
R6	Over Budget	Financial	5%	L	Keep track of expenses and Bill of Materials	Preston Pickering	Transfer. Communicate with Project Partner about budget increase and possible compromises
R7	Team cannot meet critical deadline	Schedule	10%	H	Keep track of critical deadlines in calendars and during meetings	EECS/ME Teams	Transfer. Tell the project partner and Capstone professors if the team is not projected to meet a deadline or already missed one. Ask about possible compromises
R8	Incompatible Parts	Engineering	15%	M	Thoroughly read datasheets to check compatibility. If parts are not working during implementation, try alternative parts if available	Sean Booth	Retain. Salvage the design work from the incompatible parts if possible, then either port design to a new part or redo that design

Table 2: Risks Table

2.4 References and File Links

2.4.1 References

[1] S. Wojtyła, P. Klama, and T. Baran, "Is 3d printing safe? analysis of the thermal treatment of thermoplastics: Abs, pla, pet, and nylon," *Journal of Occupational and Environmental Hygiene*, vol. 14, no. 6, pp. D80–D85, 2017. DOI: 10.1080/15459624.2017.1285489.

[2] F. L. Chan, R. House, I. Kudla, J. C. Lipszyc, N. Rajaram, and S. M. Tarlo, "Health survey of employees regularly using 3d printers," *Occupational Medicine*, vol. 68, pp. 211–214, 3 2018. DOI: 10.1093/occmed/kqy042.

[3] L. Zeng and G. M. Bone, "Design of foam covering for robotic arms to ensure human safety," presented at the 2008 Canadian Conference on Electrical and Computer Engineering, Niagara Falls, ON, Canada: IEEE, 2008. DOI: 10.1109/CCECE.2008.4564717.

[4] M. Gebler, A. J. S. Uiterkamp, and C. Visser, "A global sustainability perspective on 3d printing technologies," *Energy Policy*, vol. 74, p. 158 167, 2014. DOI: 10.1016/j.enpol.2014.08.033.

[5] M. Ratto and R. Ree, "Materializing information: 3d printing and social change," *First Monday*, vol. 17, no. 7, 2012. DOI: 10.5210/fm.v17i7.3968.

[6] J. Faludi, C. Bayley, S. Bhogal, and M. Iribarne, "Comparing environmental impacts of additive manufacturing vs traditional machining via life-cycle assessment," *Rapid Prototyping Journal*, vol. 21, pp. 14–33, 1 2015. DOI: 10.1108/RPJ-07-2013-0067.

[7] M. R. Khosravani and T. Reinicke, "On the environmental impacts of 3d printing technology," *Applied Materials Today*, vol. 20, 2020. DOI: 10.1016/j.apmt.2020.100689.

[8] C. Freund, A. Mulabdic, and M. Ruta, "Is 3d printing a threat to global trade? The trade effects you didn't hear about," *World Development Report*, 2020. [Online]. Available: https://openknowledge.worldbank.org/bitstream/handle/10986/32453/WPS9024.pdf.

2.4.2 File Links

2.5 Revision Table

Date	Revision Info
10/29/2021	Sean Booth: Set up section formatting. Reviewed and revised content.
10/29/2021	Samuel Lewis: Reformatted and fleshed out Risk Table (2.3).
10/29/2021	Preston Pickering: Added requirements section, began risks table.
10/29/2021	Garrett Hallquist: Risk Table - Added risks 1-4
11/12/2021	Garrett Hallquist: Risk Table - Added risks 5-8.
11/12/2021	Samuel Lewis: Requirements - Edited requirements for Inverse Kinematics, Quarter Cylinder Print, and Instruction Interface.
11/12/2021	Sean Booth: Updated requirements to be more broken down and fleshed out, with as much detail as is currently possible.
11/12/2021	Preston Pickering: Added risk management keywords to risks table as requested by Ingrid.
11/19/2021	Sean Booth: Edited requirements 1, 2, 3, 7, and 8 for clarity and rotated the actual page for the risks table, as per feedback.
12/2/2021	Sean Booth: Edited requirements to remove "how" details and focus on system capabilities. Verification process updates in progress.
12/3/2021	Sean Booth: Finished updating verification processes.
3/6/2022	Sean Booth: Started updating requirements to rely less on ME design.
5/6/2022	Sean Booth: Added design impact statement.

3 Top-Level Architecture

3.1 Block Diagram

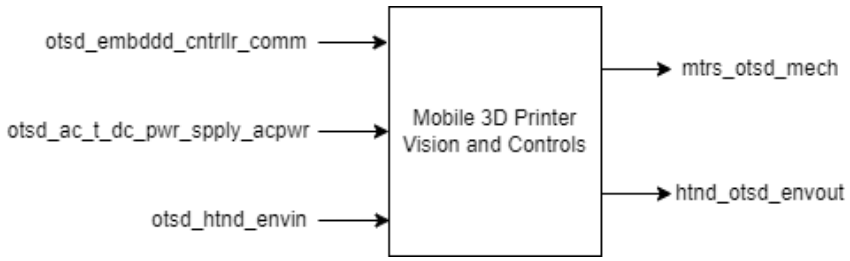


Figure 2: Black Box Diagram

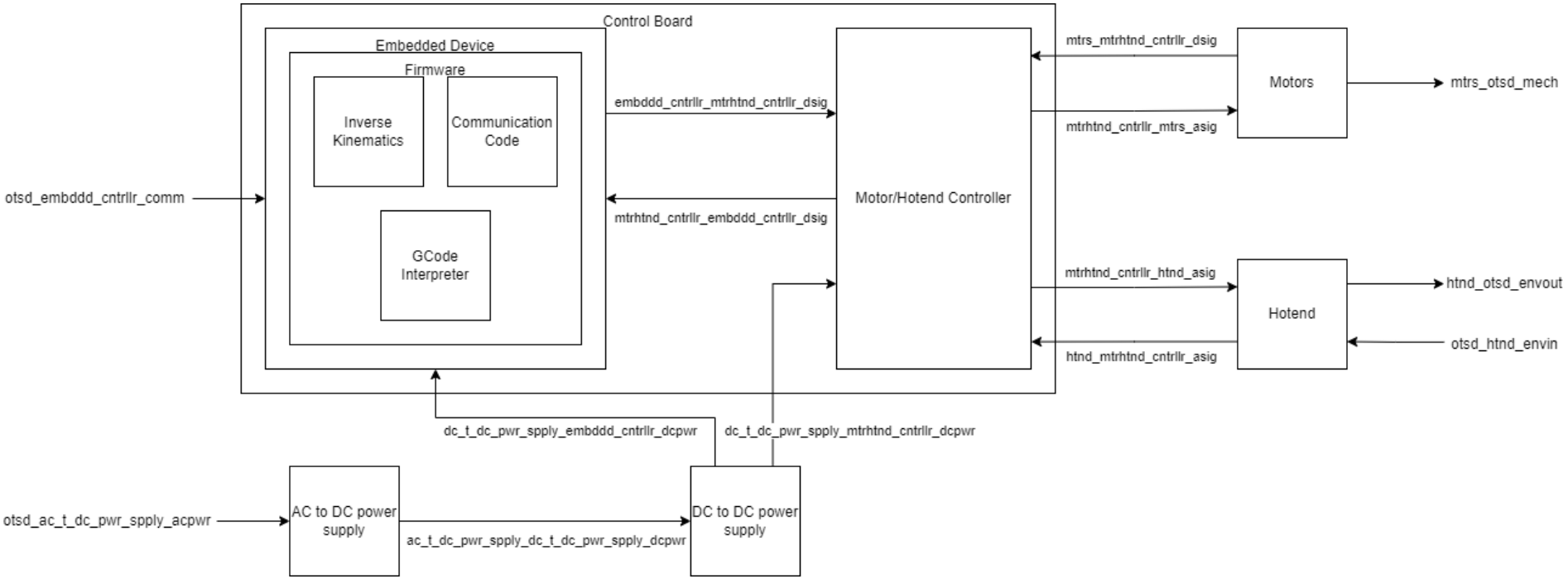


Figure 3: Block Diagram

3.2 Block Descriptions

3.2.1 Parser Block

Champion: Preston Pickering
Description: The printer will parse a CSV file and provide data to the

3.2.2 Firmware

Champion: Samuel Lewis
Description: The firmware on the micro-controller will take input from the inverse kinematics algorithm, instead of using the GCode output like you see in 3-axis printing, and translate the values into useable output to be passed to the stepper motors and hot-end. The firmware will also be in charge of monitoring the hot-ends temperature and stopping the print process if there is thermal runaway.

3.2.3 Stall Detection Code

Champion: Samuel Lewis
Description: The stall detection will measure the speed-load-angle output of each motor to see if a stall has occurred.This is required for the printer to have collision detection and an accurate homing sequence. Without encoders the system will rely completely on stall detection to calibrate itself and avoid damaging itself or the print.

3.2.4 Inverse Kinematics

Champion: Preston Pickering
Description: In order to take in a set of coordinates and move the print head to that position and normal vector from its current position and normal vector, inverse kinematics will be utilized. The current approach we are planning on is to use cyclic coordinate descent. Effectively, it cycles through all segments of the arm and uses heuristics to simulate the arm moving closer to the location. This will be done until the print head is within a tolerance of the target area. Then the motors will be moved to match the final position in the simulation. The algorithm will produce the change in angle or axis movement required to get the print head to the desired position.

3.2.5 Control Board

Champion: Garrett Hallquist
Description: The Control Board is the physical PCB that will contain the Embedded Device and the Motor/Hotend Controller. It will provide the contained blocks with the physical interface connections they need. The Control Board will be mounted on the mechanical design.

3.2.6 Embedded Device

Champion: Garrett Hallquist
Description: The Embedded Device is the microcontroller set-up that will contain all the 3D printer firmware. It will accept printer instructions from the external connection, DC power from the DC to DC power converter, and feedback from the Motor/Hotend controller.

3.2.7 AC to DC Converter

Champion: Sean Booth
Description: The AC to DC Power Supply converts a standard 120VAC 60Hz wall outlet power supply to a DC power supply. This block supplies power to the whole system.

3.2.8 DC to DC Converter

Champion: Sean Booth
Description: The DC to DC Power converter converts the DC power from the AC to DC power converter into two DC power supplies. One supply will power the Embedded Device and match the Embedded Device power specifications. The other will supply the Motor/Hotend Controller and match the Motor/Hotend power specifications.

3.2.9 Motor/Hotend Controller

Champion: Sean Booth
Description: The Motor/Hotend Controller block accepts power from the power supply and the control instructions outputted by the Embedded Device. The Motor/Hotend Controller will output power to the Motor and Hotend blocks, as well as convert the Embedded Device instructions to the desired motor signals. The Motor/Hotend Controller will send Motor/Hotend information to the Embedded device.

3.2.10 Motors

Champion: Sean Booth
Description: This block represents both the motors and the wiring that connects them to the motor controls and power. The connection of the motors to the mobile 3D printer frame will be a collaboration with the Mechanical Team. The motor connections need to meet motor specifications, and must be wired in a way that does not impede the movement of the printer.

3.2.11 Hotend

Champion: Sean Booth
Description: The Hotend block represents the electrical component of the part that heats the filament at the end of the printer arm before being deposited onto the print surface. The Hotend block measures the current temperature of the heating block, and sends that information to the Motor/Hotend Controller. The Motor/Hotend Controller will send an analog signal to the Hotend to control the temperature of the heating block.

3.3 Interface Definitions

Interface Name	Interface Properties
otsd_embddd_dvc_comm	Wired Communication: • Datarate: 9600 baud • Protocol: USB • Other: GCode Input
otsd_ac_t_dc_pwr_spply_acpwr	AC Power: • $V_{nominal}$: 120VAC • I_{peak}: 2A • $I_{nominal}$: 600mA • Other: NEMA 5-15R
otsd_htnd_envin	Environmental Input: • Temperature (Absolute): 0-300^{o}C

Interface Name	Interface Properties
embddd_dvc_mtrhtnd_cntrllr_dsig	Digital Signal: • V_{max}: 5V • V_{min}: 0V • Other: Motor/Hotend Control Signals
ac_t_dc_pwr_spply_dc_t_dc_pwr_spply_dcpwr	DC Power: • V_{max}: 25.2V • V_{min}: 22.8V • I_{peak}: 10A • $I_{nominal}$: 3A
dc_t_dc_pwr_spply_embddd_dvc_dcpwr	DC Power: • V_{max}: 5.25V • V_{min}: 4.75V • I_{peak}: 1A • $I_{nominal}$: 500mA
dc_t_dc_pwr_spply_mtrhtnd_cntrllr_dcpwr	DC Power: • V_{max}: 25.2V • V_{min}: 22.8V • I_{peak}: 9.8A • $I_{nominal}$: 2.9A
mtrhtnd_cntrllr_embddd_dvc_dsig	Digital Signal: • V_{max}: 5V • V_{min}: 0V • Other: Motor/Hotend Feedback
mtrhtnd_cntrllr_mtrs_asig	Analog Signal: • V_{max}: 25.2V • V_{range}: 0-24V • Max Frequency: 1kHz • Other: 4-Wire Motor Drive Signal

Interface Name	Interface Properties
mtrhtnd_cntrllr_htnd_asig	Analog Signal: • V_{max}: 25.2V • V_{range}: 0-24V • Other: Hotend Drive Signal
mtrs_otsd_mech	Dynamic Mechanical Connection: • RPM: 0-300
mtrs_mtrhtnd_cntrllr_dsig	Digital Signal: • V_{max}: 5V • V_{min}: 0V • Other: Encoder Return Signal
htnd_otsd_envout	Environmental Output: • Temperature (Absolute): 190-270^{o}C
htnd_mtrhtnd_cntrllr_asig	Analog Signal: • V_{max}: 5.25V • V_{range}: 0-5V • Other: Temperature Return Signal

Table 3: Interface Definitions Table

3.4 References and File Links

3.4.1 References

3.4.2 File Links

3.5 Revision Table

Date	Revision Info
11/19/2021	Preston Pickering: Began section 3 and added block diagram image.
11/19/2021	Samuel Lewis: Began block diagram descriptions and added description 2 and formatted block diagram.
11/19/2021	Garrett Hallquist: Added content to the block descriptions 1,3,4, and 6. Added Interface Definitions table with 8 entries.
11/19/2021	Sean Booth: Added content to block description 5, edited and expanded interface definitions.
12/3/2021	Sean Booth: Updated block diagram and interface definitions to reflect current state of project and conform with naming requirements.
12/3/2021	Garrett Hallquist: Updated block definitions based on updated block diagram.
12/3/2021	Samuel Lewis: Added block description for Firmware and updated description of Communication code.

4 Block Validations

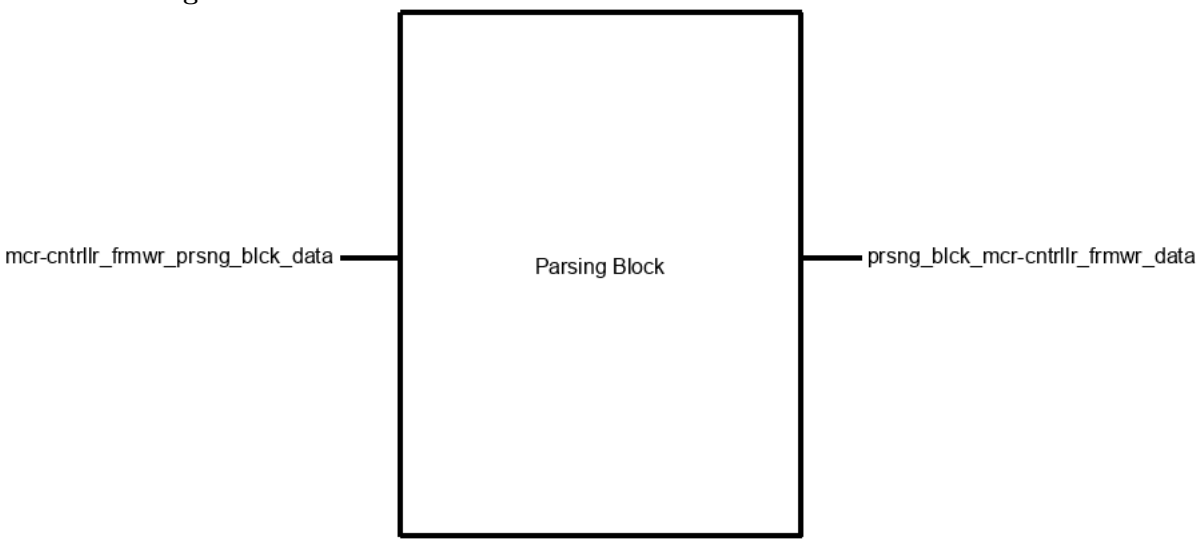
4.1 Parser Block

4.1.1 Block Overview

I (Preston Pickering) will be producing C code to act as an interface between the data provided by the user and the actual running of the printer. The code will take the user input as a csv file and have the printer begin moving motors in order to make that happen. This block should be fairly straightforward, relying on a csv library to do most of the heavy lifting. Each row in the csv file will contain six values. The first three will represent the coordinates the printer will need to move to. These coordinates are all relative to the homing position of the printer. The next two will represent the surface normal the printer needs to be at when it reaches the point. The final value will be either a one or a zero to represent whether the plastic extruder will be running. This format has been agreed upon with the project partner. Earlier in the project it was expected that we would be parsing GCode, which would have been a much more difficult task. Because of this, this block will be slightly extended to handle the main function of the printer's program as well.

4.1.2 Block Design

4.1.2.1 Block Diagram



4.1.2.2 Pseudocode

```
Main()
    //Setup
    Pull data from CSV file
    Pass CSV data into CSV reading library to produce a 2D array
    Move the printer the the home position
    Create a new printer struct representing the printer in the home position
    Set all motor angles in the printer struct to zero

    //Run print job
    For row in 2D array
        Call inverse_kinematics() on the position and normal info (See IK block)
        If the inverse_kinematic function errors:
            Stop extruding if extruding
            Print an error message
            End the program
        If the extrusion bit is set, begin extrusion. If not, stop extrusion.
        For each motor in the printer struct
            Call functions to move motors to calculated position (Firmware block)
            If the motor movement functions error:
```

```
            Stop extruding if extruding
            Print an error message
            End the program

//Shutdown
End extrusion
```

4.1.3 Block General Validation

Parsing the CSV file will be straightforward, as it is a standardized format and the format is known and agreed upon. A library can be used to turn it into an array to iterate over. My current choice for a parsing library is the libcsv library (see file links). The first three values will be floats showing the position of the next point (relative to the homing position). The next two will be floats showing the surface normal in sphere coordinate angles in radians. The last value will be a boolean for extrusion. One thing I am worried about is the actual path the print head will be taking while extruding. The IK function is capable of producing the motor angle coordinates for each movement from point to point. However, I am worried about the timing of the movements when extruding plastic. I do not have enough experience with this field to know whether simply attempting to move each motor sequentially and waiting to hear back from it can be accomplished in enough time to prevent artifacts on the printed object. Homing is another concern, as I do not know how we will accomplish this. I have not been working with the motors and have not worked with motors in the past. I get the impression that we have a plan for this, but I do not yet know what the plan is. Both of these issues are things I am going to be bringing up with my team and the mechanical engineering team at our next meeting. In general, I am confident in my design for the code itself but not in how the code will function as part of a greater whole.

4.1.4 Block Interface Validation

prsng_blck_mcr-cntrllr_frmwr_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Messages: Whether to move motors and by how much	The program will need to know to move the motors in order to actually print anything.	The actual movement of the motors and determining of how much will be done by the motor firmware and inverse kinematics blocks, respectively.
Messages: Whether there have been any errors in attempting to move motors	This will help prevent damage to the printer and otherwise make the design more resilient.	Error handling is not yet well defined in my design. I expect both the inverse kinematics function and the firmware to have errors. This block will handle them, but the errors are still undefined.
Protocol: Function calls to firmware code	It's a straightforward way of handling the input data	Numerous function calls will be made in the course of using my design.

mcr-cntrllr_frmwr_prsng_blck_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Messages: Whether to extrude plastic	This is important information which will be required in order to print anything.	This information will be in the CSV, as provided by the project partner.
Messages: Sequence of target locations and normals	This is important information which will be required in order to print anything	This information will be in the CSV, as provided by the project partner.
Protocol: CSV file	Specified by project partner	I will use a CSV library to parse the CSV.

4.1.5 Block Testing Process

A test program will be created to ensure the code is working correctly. I will use the actual CSV provided by our project partner to test the program.

1. Ensure CSV file is at the correct file path
2. Run the test code and ensure all test cases are met

4.1.6 References and File Links

No references

4.1.6.1 File Links

- CSV Parser code: csv_parser.c

4.1.7 Revision Table

Date	Revision Info
2/4/2022	Preston Pickering: Worked on all sections in order to prepare rough draft
2/18/2022	Preston Pickering: Added info on CSV format as requested by peer reviews. Added pseudocode as requested by peer reviewers
2/18/2022	Preston Pickering: Added note about the poor state of the interfaces.
2/18/2022	Preston Pickering: Added source for the csv parsing library which will be used.
2/18/2022	Preston Pickering: Note - no instructor or TA feedback was given on the rough draft
5/1/2022	Preston Pickering: Transferred block validation to project document, adapted to meet current realities of the project

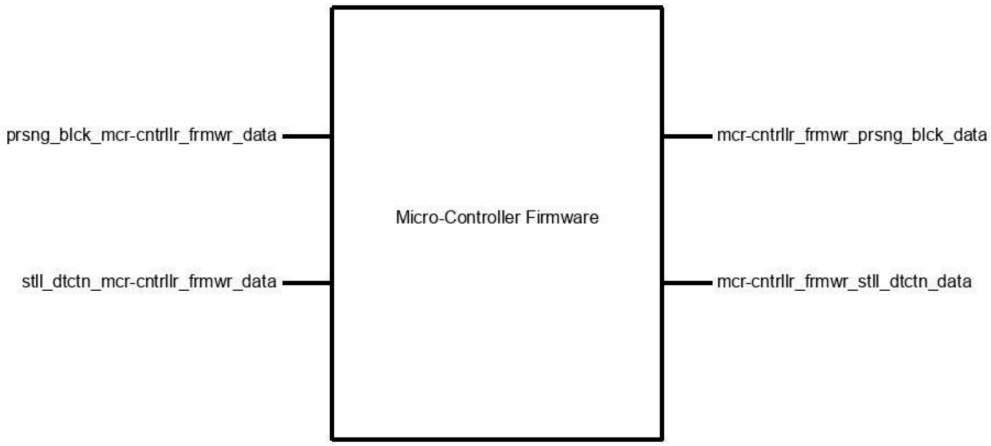
4.2 Firmware Block

4.2.1 Block Overview

The microcontroller Firmware block for this project will be done by Samuel Lewis. This block will be Arduino code that is run on our STM 32 Nucleo-64 microcontroller that will control the main print loop for the printer. This block needs to take in input from the inverse kinematics over serial, this data will indicate the desired positions and rotations of the motors by providing the difference from the known current position to the desired state. These values will be translated into step amounts for each motor taking into account the motor used and the joint type. A main program loop will be run for these desired commands while using the stall detection library in order to detect collisions during the print process. There will also be an callable homing sequence that is used before every print and is used to calibrate the motors and joints so that they are in a known position for the inverse kinematics calculations. The stall detection calls that are needed to indicate a successful homing sequence will only gather data on specific step intervals. These intervals will be different depending on the joint type and the motor speed.

4.2.2 Block Design

4.2.2.1 Block Diagram



4.2.2.2 Firmware Loop Pseudocode

```
print_sequence(serial)
    Read in message from buffer
    See if message is in the correct format
    If message is move command
        Confirm data
        Run move command
        Request more data
    If message is homing request
        Run homing sequence
        Confirm homing complete
    If message is temp command
        Set temperature of hot-end
    If message is debug
        Enter debug mode
        Request debug command
    else
        Report bad message
```

4.2.2.3 Movement Pseudocode

```
move_command(angle changes)
    Convert angles to step amounts
    Set direction for each motor
    Calculate step delay and torque for each motor
    While the movement is not done
        If it is time to move one of the 5 motors
            Move that motor
            If it is time to check for stall
                If there is a stall on that motor
                    Return stall
        If all motors are done
            Break out of loop
    Return successful movement
```

4.2.2.4 Homing Pseudocode

```
homing_command()
    Set direction for each motor to homing motor
    Make list of all motors set to running
    While homing is not done
        If it is time to move one of the 5 motors
            Move that motor
            If it is time to check for stall
                If there is a stall on that motor
                    Set that motor to not running
        If all motors are not running
            Break out of loop
    Return successful homing
```

4.2.3 Block General Validation

Using the required movement from the inverse kinematics the program will calculate the amount of steps required for each motor. This will be different for the rotational joints, the prismatic joints and the extruder. All joints will be stepped at speeds such that they all reach the desired position at the same time. There will also be a homing sequence at the beginning of each print where the motors are not given a desired position but are given a direction to move in infinitely. In this sequence a motor will only be stopped once it has stalled. This will be flagged using the stall detection algorithm library

The firmware will also be responsible for the Arduino side of communication. Sending requests and status updates to the computer running the Inverse Kinematics. To minimize the communication over serial, which risks blocking the Arduino loop, the IK will only need to start sequences that are pre-programmed onto the board such as the homing and print sequences. The print sequence will require intermittent communication while running to allow for a stream of data to the board minimizing the amount of memory and processing time it must dedicate to storing long paths.

4.2.4 Block Interface Validation

mcrcntrllr_frmwr_stll_dtctn_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Other: readAnalog()	Stall detection needs to be able to read the voltage from the analog SLA PIN on each of the motor drivers.	This will be a function for reading the input of readAnalog() and returning a digital value usable by the stall detection.
Other: setSLAGainDefault()	Stall detection algorithm requires the gain to be at a readable amount depending on the voltage and speed of the motor.	This will be a function for outputting a value to the pins connected to the motor driver when called by the stall detection algorithm.
Other: setSLATransparencyOff()	Stall detection algorithm requires the transparency of the SLA pin to be set to off when reading the back EMF.	This will be a function for outputting a value to the pins connected to the motor driver when called by the stall detection algorithm.

mcrcntrllr_frmwr_invrs_knmtcs_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Messages: Angle values, matrices and data describing the printer structure, out-of date link position matrices.	This is all the information necessary to perform the function of the block. The link position matrices hold some of that data, but by only checking the motor angles this removes those being out of date as a point of failure.	This data is stored in one of the data structures being passed into the kinematics functions.
Messages: Coordinates and normal vector of the target point.	This information is required, as it is the target point which the printer is trying to move to.	This data is being stored in one of the data structures being passed into the inverse kinematic function.
Protocol: Arm data structure pointer, two cglm library vector 3s.	The arm data structure was constructed to store all the necessary data about the arm across the entire program. Passing it by reference is more efficient than passing it by value. CGLM was selected as a matrix math library due to my familiarity with GLM. Two vectors are needed to represent the normal vector and the target point.	These data structures are in the function definitions.

mcrcntrllr_frmwr_mtrhtnd_cntrllr_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Messages: Translated step amount for rotational joints.	Motors need to be stepped until they reach the desired rotation.	We will get the input of desired positions from the inverse kinematics.
Messages: Translated step amount for prismatic joints.	Motors need to be stepped until they reach the desired rotation.	We will get the input of desired positions from the inverse kinematics.
Messages: Extruder motor step amount.	Motors need to be stepped until they reach the desired rotation.	We will get the input of desired positions from the inverse kinematics.

invrs_knmtcs_mcrcntrllr_frmwr_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Messages: The arm structure is not changed outside of the angles and link matrices.	The arm data structure contains information relating to the structure of the arm, which will not be changing during its operation.	Some of the variables involved will be set as constants, preventing them from being written to once they are assigned.
Messages: The motor angles of the arm changed to be those required to produce the target point and normal vector when put through forward kinematics.	This is the entire point of the block. The purpose of the block is to translate a change in cartesian space to a change in angle space.	At this point with the design of the robot known, it should just be a matter of simply calculating the sphere coordinates of the normal vector and applying those to the rotation joints, and fairly simple math for the prismatic joints. My engineering judgment is that it ought to work.
Protocol: Editing the arm structure found at the pointer passed as an input	This data structure was constructed to store all the necessary data about the arm across the entire program. Passing it by reference is much more efficient than passing it by value.	The functions are void functions, and output nothing. They edit the data through the provided pointer.

stll_dtctn_mcrcntrllr_frmwr_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Messages: Stall Alert during print phase when colliding with unknown object.	This is needed to deactivate the entire system in the event of a collision.	Every step a stall detection algorithm will be called and will automatically trigger if there is a motor has collided with something.
Messages: Homing Completion Status Alert when motors are successfully or unsuccessfully homed.	All motors must be in a known position in order to print correctly. If the home is unsuccessful the print must be stopped.	If a motor has not been homed it will have to be accessed by the team and corrected so the printing phase can commence.
Protocol: Code will be downloaded over UART connection.	This is the only way to load Arduino code onto this microcontroller.	A .ino file is loaded onto the STM 32 microcontroller that will run this code.

4.2.5 Block Testing Process

1. The firmware will send a data request message to the IK system over serial.
2. The IK will read this message and return the requested information.
3. This returned data will be checked against the expected output.
4. The firmware will now calculate and output the correct step amount for each of the motors.
5. A stall check will then be requested from the stall detection system and a result will be returned.
6. The motors will be run to their homing position so we know that a stall should have occurred.

7. We will then check that the motors were run.
8. We will check that a stall flag was sent.
9. And we will also check that all motors were stopped upon receiving the flag.

4.2.6 References and File Links

4.2.6.1 File Links

- Main Firmware: firmware.ino
- Motor Control Library: printer_control.cpp & printer_control.h
- USB Communication Library: usb_lib.cpp & usb_lib.h

4.2.7 Revision Table

Date	Revision Info
2/2/2022	Samuel Lewis: Added template sections
2/3/2022	Samuel Lewis: Filled out block overview
2/3/2022	Samuel Lewis: Added interfaces from past blocks
2/3/2022	Samuel Lewis: Finished all interface descriptions
2/4/2022	Samuel Lewis: Finished references and file links
2/4/2022	Samuel Lewis: Finished general validation
2/16/2022	Samuel Lewis: Included numbered steps and info on microcontroller to plan as per feedback
2/16/2022	Samuel Lewis: Added pseudocode as per feedback
2/16/2022	Samuel Lewis: Fleshed out Overview section

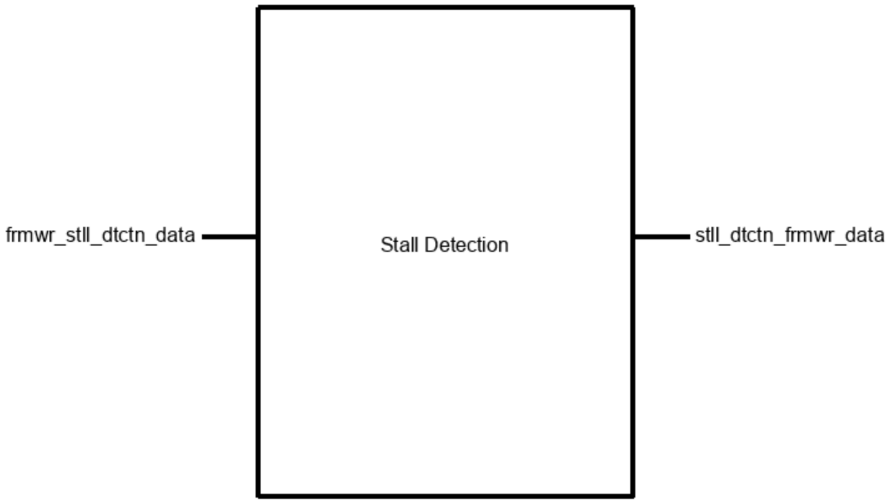
4.3 Stall Detection Block

4.3.1 Block Overview

I (Samuel Lewis) am responsible for the Stall Detection block of the project. This block will be Arduino and C++ code running on our STM 32 Nucleo-64 micro-controller that will monitor the data from the motor-drivers to determine when a motor has stalled. These stalls occur when the motor cannot accomplish a step command despite receiving the correct voltage. This will be reflected in the speed-load-angle output which gives us the back-EMF voltage. The voltage we measure after a step will reflect the speed it is moving so we can monitor this to see when a motor has stopped when it should be moving.

4.3.2 Block Design

4.3.2.1 Block Diagram



4.3.2.2 Stall Detection Queue Pseudocode

```
push_voltage(voltage, motor_driver_log)
    Remove next open slot data from average
    Clear next slot
    Add voltage reading to next position
    Calculate average change
    add change onto average
    If new voltage is below stall line
        Add stall trigger
        If stall triger is high enough to indicate stall
            return stall on motor
    else
        reset trigger
```

4.3.3 Block General Validation

The two cases of stalls that are interesting to the team are End-of-Line (EOL) and collision base stalls. EOL stalls happen when a motor reaches its limit of movement in one direction and can no longer take a step. This will often be triggered intentionally as a method to home a joint on the printer by starting from a known point in the joint's range of motion. If this happens when not in a homing state it will be treated as a collision with the EOL. A collision with an object should not happen outside of the homing state so any motor stalls during printing will trigger a pause to the main function and stop any further motor movement.

This block will act as a sort of complex queue data type that monitors the values being pushed into it and compares them to the averages of the voltages currently inside of the queue. This will allow us to see any sudden changes while not falsely triggering on the normal operating variance or the gradual change in resistance as a motor rotates to new position that is effected by gravity differently. This change angle relative to gravity is even more important for the rotational motors as each movement will directly cause a significant change.

4.3.4 Block Interface Validation

stll_dtctn_frmwr_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Messages: Stall Alert during print phase when colliding with unknown object.	The print sequence returns a value indicating if the print sequence ended due to a stall on one of the motors or due to the end of the print sequence. If the sequence was ended due to a stall, the id of the first stalled motor will be returned as the value.	Every step a stall detection algorithm will be called and will automatically trigger if there is a motor has collided with something.
Messages: Homing Completion Status Alert when motors are successfully or unsuccessfully homed.	The homing sequence returns a value indicating whether it ended because all motors successfully reached their home state, or if the sequence ended without homing all motors. An integer is returned, 0 for failure and 1 for success.	If a motor has not been homed it will have to be accessed by the team and corrected so the printing phase can commence.
Protocol: Code will be downloaded over UART connection.	This is what the AMIS30543 requires to use the arduino library.	A .ino file is loaded onto the STM 32 microcontroller that will run this code.

frmwr_stll_dtctn_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Other: readAnalog()	Reads the analog signal of the Speed Load Angle PIN on the motor driver. Returns a number of 0 to 1023 associated to the voltage of 0 to 5V. This is being simulated using a random data generator until we have set up an Arduino environment.	This will be a function for reading the input of readAnalog() and returning a digital value usable by the stall detection.
Other: setSLAGainDefault()	Method of class AMIS30543. This function clears the SLAG bit and sets the gain on the SLA pin of the motor to the default value of 0.5. This function is temporarily shown by data similar to that generated with and without a default gain.	This will be a function for outputting a value to the pins connected to the motor driver when called by the stall detection algorithm.
Other: setSLATransparencyOff()	Method of class AMIS30543. This function clears the SLAT bit which turns off the SLA transparency. This makes the SLA signal only sample once per coil zero current crossing and simplifies our input.	This will be a function for outputting a value to the pins connected to the motor driver when called by the stall detection algorithm.

4.3.5 Block Testing Process

1. Each motor will be given an individual move command in the positive direction (2000 steps for prismatic joints and 150 steps for rotational joints) with nothing blocking their path.
2. Each motor will be given an individual move command in the negative direction (2000 steps for prismatic joints and 150 steps for rotational joints) with nothing blocking their path.
3. Both of these move commands should have been completed with no errors being thrown of stalls being flagged.
4. Now each motor will be given the same commands but an object will be placed in its way.
5. When each motor collides with the object they should stop moving immediately and sent a stall flag that indexes the correct motor.

4.3.6 References and File Links

Motor Control Library: stall_detection.cpp & stall_detection.h

4.3.7 Revision Table

Date	Revision Info
1/12/2022	Samuel Lewis: Created file and labeled sections
1/13/2022	Samuel Lewis: Started interface validation
1/20/2022	Samuel Lewis: Finished block design and overview
1/20/2022	Samuel Lewis: Added files
1/24/2022	Samuel Lewis: Removed all old current sensing information. Added description of new method
1/29/2022	Samuel Lewis: Added new interfaces

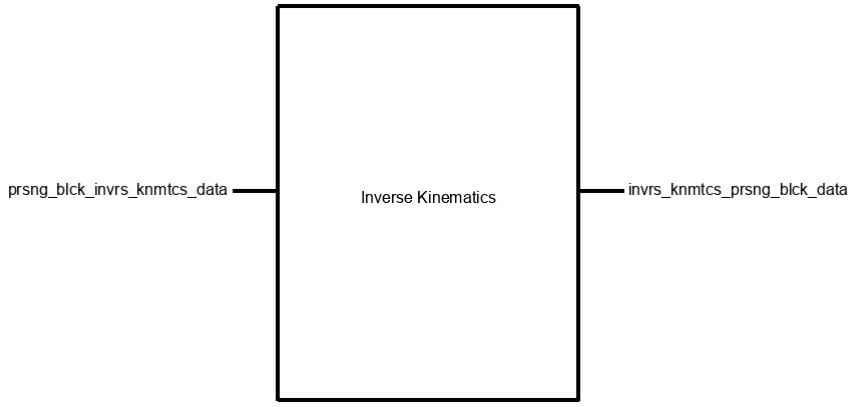
4.4 Inverse Kinematics Block

4.4.1 Block Overview

I (Preston Pickering) will be producing C code to act as an interface between the data provided by the user and the actual running of the printer. The code will take the user input as a csv file and have the printer begin moving motors in order to make that happen. This block should be fairly straightforward, relying on a csv library to do most of the heavy lifting. Each row in the csv file will contain six values. The first three will represent the coordinates the printer will need to move to. These coordinates are all relative to the homing position of the printer. The next two will represent the surface normal the printer needs to be at when it reaches the point. The final value will be either a one or a zero to represent whether the plastic extruder will be running. This format has been agreed upon with the project partner. Earlier in the project it was expected that we would be parsing GCode, which would have been a much more difficult task. Because of this, this block will be slightly extended to handle the main function of the printer's program as well.

4.4.2 Block Design

4.4.2.1 Block Diagram



4.4.2.2 Data Structures Even though the 3D printer is not a robot arm proper, from the perspective of the kinematics functions there is little difference. It can be modeled as an arm consisting of links and different types of joints.

The arm data structure will need to store

- An ordered array of link structures. The last link will be the end effector.
- An ordered array of motor structures, corresponding to the links.

Each link will need to store

- The relative position of the end of the link relative to the start of the link when the printer is homed, stored as a matrix. I call this the "constant" matrix as it represents the unchanging structure of the printer.
- The current relative position of the end of the link relative to the start of the link, stored as a matrix. This is the "current" matrix as it is the one which will be changed.
- The current position of the end of the link relative to the origin, defined as the base of the first link while in the home position. This is the "absolute" matrix.
- The type of joint the link has with the previous link, either prismatic or rotational
- A unit vector describing the movement of the end of the joint, either the direction of movement for a prismatic joint or the axis of rotation for the rotational joint.
- The ratio between the motor's change in angle and the link's movement.

Each motor will need to store

- The current angle the motor is at, stored relative to the home position of 0.
- The maximum and minimum angles the motor can safely be at, also stored relative to the home position.

4.4.2.3 Forward Kinematics Pseudocode

```
forward_kinematics(Pointer to arm data structure)
    For each link/motor in the arm
        If the link is prismatic
            Set the link's current matrix to be its constant matrix translated
            along the links axis by the angle of the motor times the movement
            ratio
        Otherwise
            Set the link's current matrix to be its constant matrix rotated
            around the link's axis by the angle of the motor times the movement
            ratio
    For each link in the arm
        Set the link's absolute matrix to be its current matrix times the
        previous link's absolute matrix. For the first link, just use the
        current matrix.
```

4.4.2.4 Inverse Kinematics Pseudocode

```
inverse_kinematics(Pointer to arm data structure, vector3 target_position,
vector3 normal)
    Perform forward kinematics to get the current position of the print head
    Handle normal vector
        Calculate the sphere coordinates of the normal vector
        Use those coordinates, along with the movement ratios, to change the
        motor angles for the two rotational joints.
        If the angles are outside of the motor's range, clip them
    Handle position
        Perform forward kinematics to be able to get the length and x-y-z offset
        for the last two joints, effectively pretending they are a single
        vector.
        Calculate difference between current and target print head positions for
        all axes.
        Move each axis, taking movement ratios into account
        If the angles are outside of the motor's range, clip them
```

4.4.3 Block General Validation

The general case of inverse kinematics is a very difficult problem which I neither have the math background for nor the time to solve. Fortunately, I am reasonably sure I can tailor the algorithm to the design of the printer and simplify things immensely. The printer has 3 prismatic (straight line) axes as a base, with two rotational links at the end. This means as long as I can implement the comparatively simple 2-link IK problem, I should then be able to treat the rotational section of the arm as a single vector and move the 3-axis to get the tip into the right location.

The 2-link subproblem is complicated by the fact that, due to printing on a curved surface, I will need to have an associated normal vector at each point along the print surface. A normal vector is a vector which is perpendicular to the surface. However, this also affords me additional information. I am hoping that I can simply do a bit of trigonometry to figure out the relationship between the spherical coordinates of the normal vector and the angles the motors are at. Then I can simply have it converted in order to solve the 2-link IK problem for the rotational joints at the end of the assembly. This ought to work as it will be able to achieve both the normal vector and the required position, though it has the downside of likely needing to perform forward kinematics between the rotational and prismatic portions of the process. This method was chosen due to the lack of time and understanding needed to implement a general IK solving algorithm.

If this solution for the 2-link problem does not work, I expect to attempt using the Jacobean method [1]. This method relies on using Jacobian matrices to relate the change in the angle to the change in the cartesian coordinates in a local area as a heuristic, and iterating by updating the position of the end effector and the heuristic. Over winter break I re-taught myself enough multivariable calculus to feel confident attempting this method on the smaller two-link. The reason it was not selected as the first attempt is that, while a general solution, it does take more understanding and is more complex to program.

4.4.4 Block Interface Validation

invrs_knmtcs_prsng_blck_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Messages: The arm structure is not changed outside of the angles and link matrices	The arm data structure contains information relating to the structure of the arm, which will not be changing during its operation.	Some of the variables involved will be set as constants, preventing them from being written to once they are assigned.
Messages: The motor angles of the arm changed to be those required to produce the target point and normal vector when put through forward kinematics.	This is the entire point of the block. The purpose of the block is to translate a change in cartesian space to a change in angle space.	At this point with the design of the robot known, it should just be a matter of simply calculating the sphere coordinates of the normal vector and applying those to the rotation joints, and fairly simple math for the prismatic joints. My engineering judgment is that it ought to work.
Protocol: Editing the arm structure found at the pointer passed as an input	This data structure was constructed to store all the necessary data about the arm across the entire program. Passing it by reference is much more efficient than passing it by value.	The functions are void functions, and output nothing. They edit the data through the provided pointer.

prsng_blck_invrs_knmtcs_data		
Interface Property	Why is this interface of this value?	Why do my design details for this block meet or exceed these properties?
Messages: Angle values, matrices and data describing the printer structure, out-of date link position matrices	This is all the information necessary to perform the function of the block. The link position matrices hold some of that data, but by only checking the motor angles this removes those being out of date as a point of failure.	This data is stored in one of the data structures being passed into the kinematics functions.
Messages: Coordinates and normal vector of the target point	This information is required, as it is the target point which the printer is trying to move to.	This data is being stored in one of the data structures being passed into the inverse kinematic function.
Protocol: Arm data structure pointer, two cglm library vector 3s	The arm data structure was constructed to store all the necessary data about the arm across the entire program. Passing it by reference is more efficient than passing it by value. CGLM was selected as a matrix math library due to my familiarity with GLM. Two vectors are needed to represent the normal vector and the target point.	These data structures are in the function definitions.

4.4.5 Block Testing Process

A simple benchmark testing program will be written, checking the results of the arm against a number of cases, such as attempting to go out of bounds from the arm's limitations, checking its answers against trivial and non-trivial known results, and ensuring the program is working as expected. The prsng_blck_invrs_knmtcs_data interface is more or less baked into the program and will not need to be tested. The invrs_knmtcs_prsng_blck_data interface will need more rigorous testing. In particular, the testing program will need to ensure that the resulting point and normal vector are as close to the targets as possible. The elements of the arm structure not meant to change will also need to be checked before and after to ensure they have not changed.

4.4.5.1 Testing Procedure

1. The testing program will be run, and if all results come out as expected, the code works.

4.4.6 References and File Links

4.4.6.1 References

[1] A. Sears-Collins. "The ultimate guide to inverse kinematics for 6dof robot arms." (2020), [Online]. Available: https://automaticaddison.com/the-ultimate-guide-to-inverse-kinematics-for-6dof-robot-arms/#Inverse_Kinematics_Using_the_Pseudoinverse_Jacobian_Method_Numerical_Approach.

4.4.6.2 File Links

- Printer Data Structure: printer_struct.h
- Kinematics Code: kinematics.c

4.4.7 Revision Table

Date	Revision Info
1/6/2022	Preston Pickering: Created document, added section headers, added content for block overview, and general validation
1/7/2022	Preston Pickering: Created block diagram, edited previously added content, added interface validation, testing process, and file links
1/19/2022	Preston Pickering: Added interface information (as requested by Ingrid), edited forward kinematics pseudocode to reflect current design, added links.
1/19/2022	Preston Pickering: Refined interfaces
1/20/2022	Preston Pickering: Added IK pseudocode, added references. Added links to code.
3/6/2022	Preston Pickering: Transferred block validation to project document, edited interface names to reflect current design
5/6/2022	Garrett Hallquist: fixed reference formatting in latex.

4.5 Motor/Hotend Controller

Champion: Sean Booth

4.5.1 Overview

This block controls two major components of the system: the motors and the hotend. It takes in power from the power supply, and control signals from the embedded device, and turns this into signals capable of driving a stepper motor and controlling the hotend. It is also responsible for measuring return signals from the hotend and motor controllers. The block will contain multiple motor driver daughter boards that will be purchased off the shelf, eventually seated into a custom PCB which will also contain a MOSFET for powering the hotend, and any signal processing/-multiplexing necessary to return driver and temperature signals to the embedded device. For the purpose of validation, only a single motor driver instance will be tested, as the final block will simply feature repeated copies of the same motor driver, with some shared pins.

4.5.2 Design

At the black box level, this block takes three inputs and drives three outputs. It takes in digital signals from the embedded device, which is a collection of signals represented under one input. These signals will configure and control the motor drivers, as well as control the hotend MOSFET. It also takes in 24V DC power from the power supply, used to power the motor driver boards and the hotend MOSFET. The hotend will be returning data in the form of an analog voltage from a thermistor, set up in a voltage divider. The outputs are much simpler than the inputs. The block will drive the motors through individual 4-wire analog signals, direct from the motor driver boards. The block will also output a digital signal to drive the hotend through the use of a MOSFET and PWM. Finally, the block will output some return signals to the embedded device, such as SLA (speed and load angle) data from the motors and temperature readings from the hotend. The black box diagram can be see in Figure 4 and the schematic for the block (single motor unit) in Figure 5.

In designing this block, two major parts were chosen: the AMIS-30543 motor driver carrier board from Pololu, and the PSMN022-30PL N-channel MOSFET. Datasheets for both devices and further information can be found in 4.5.6.2 below. This block acts primarily as an interconnect for the AMIS-30543 boards, but also carries the MOSFET required for driving the hotend, and a resistor divider setup that will read a thermistor attached to the hotend to determine its temperature. Because this block will have the logic level voltage of the embedded device for reference and uses it for the voltage divider, the temperature reading will be ratiometric, which comes with some benefits.

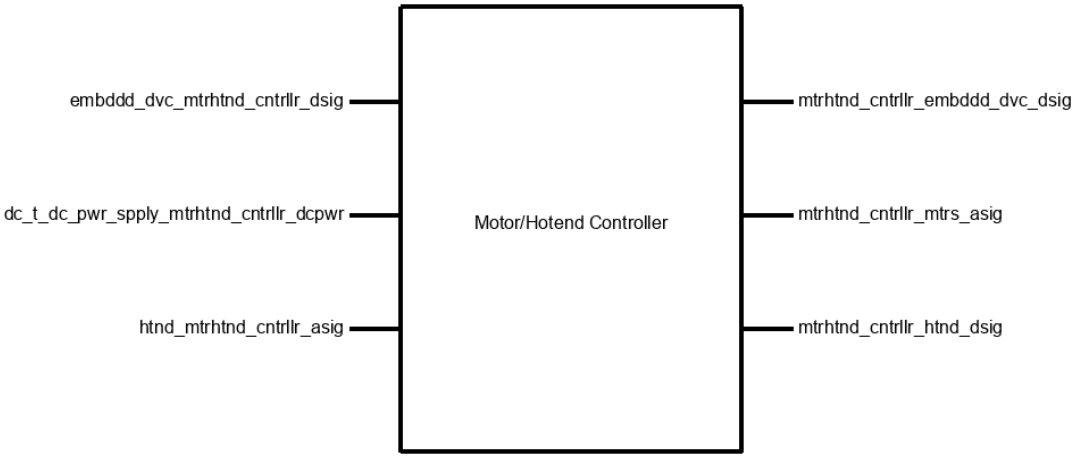


Figure 4: Motor/Hotend Controller Black Box Diagram

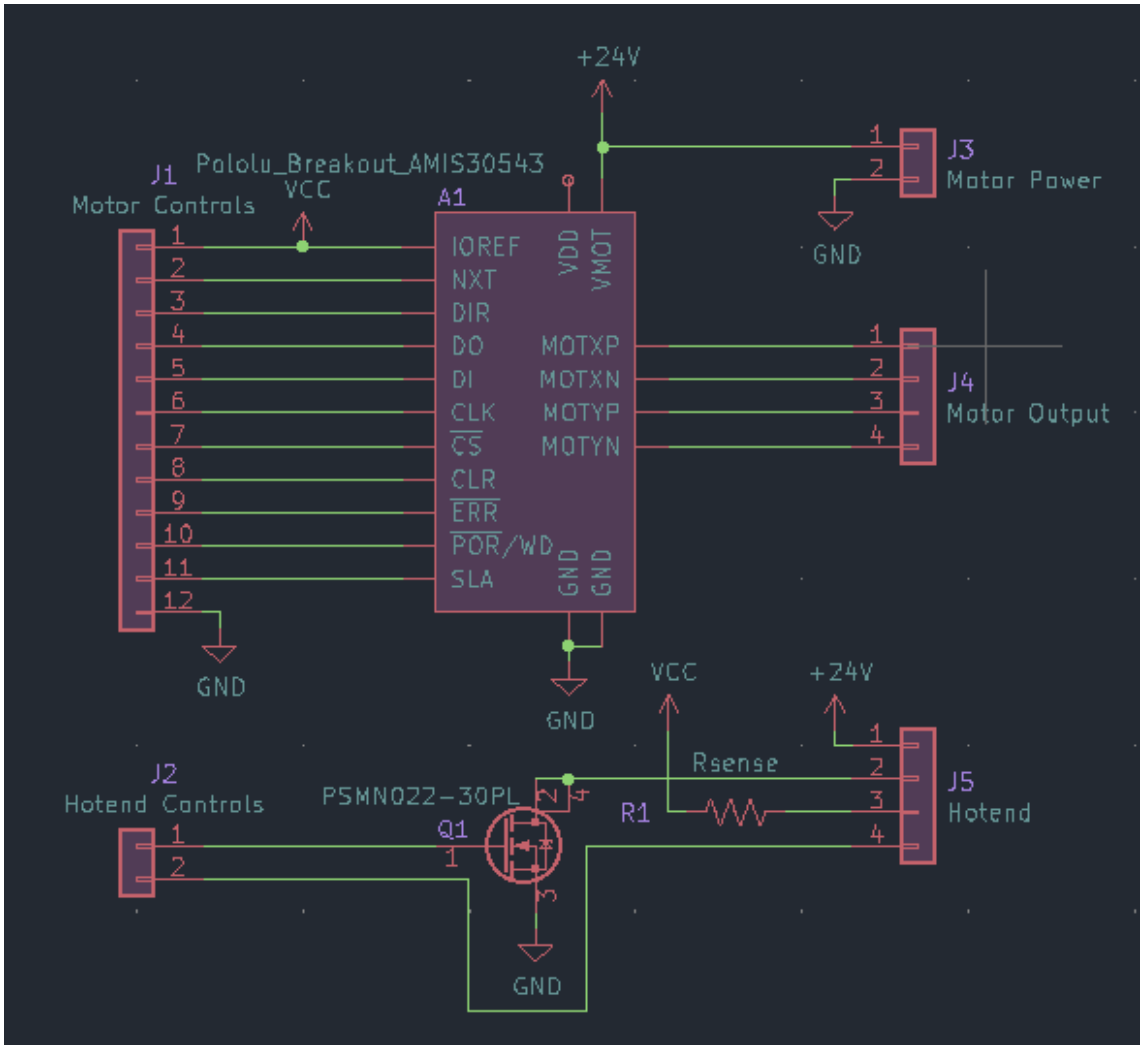


Figure 5: Motor/Hotend Controller Schematic

4.5.3 General Validation

Given what this block is supposed to do (take signals and drive motors/hotend), this design should prove to be more than sufficient for the task. Most of the design is comprised of the off the shelf motor driver boards from Pololu (AMIS-30543), which are capable of supplying 1.8A per coil to the motors (more if heatsinks or other cooling are used), and can operate on supply voltages anywhere from 6V to 30V. They also support logic levels from 2.5V to 5.5V, which should cover 3.3V and 5V devices well. They are in decent stock given current conditions, and so acquiring six units should be reasonable. Given the complexity of motor drivers in both design and layout, using an off the shelf part was deemed the best course of action for this part of the block.

Of course, a concern that comes with these is the fact that there will be up to six of them in the final block (only one is going to be tested for simplicity), meaning that with six two-phase motors, the potential total current draw could be 21.6A. However, that current draw will almost certainly not be seen by the power supply, as the motor voltage ratings will almost certainly be much lower than the supply voltage. As an example, a two-coil stepper motor with a voltage rating of 2.8V and a current rating of 1.7A per coil (reasonably typical values) will draw about 4.8W of power. Assuming high efficiency, this means a load of about 0.2A presented to the power supply per coil. With six two-coil motors, this becomes 2.4A. Given that the max supply current the block has to work with is 9.8A, the motor drivers should not draw more power than the block can handle even under the worst case scenario of all six drivers operating both coils at max current.

These drivers also fulfill the control and feedback requirements of the other system blocks. They take SPI signals from the embedded device to configure their internal settings, which will require a

large number of chip-select pins unless an alternate solution such as multiplexing or daisy chaining is used. For this unit validation, this will not be a concern, but this limitation of SPI may influence the full design, or the design of other system blocks. In any case, this SPI communication provides a number of benefits, such as being able to set current limits for individual motors in software with ease, and being able to read back any error information. The driver also provides some other useful feedback in the form of the SLA (speed and load angle) line, which can be used for stall detection, something necessary for proper operation of the mechanical hardware.

Moving away from the driver and onto the subject of the MOSFET, the chosen part here is the PSMN022-30PL, a pretty standard N-channel MOSFET. First of all, it meets the power requirements, being able to handle up to 30V and 30A. For a typical 24V 40W hotend heater core, it can be expected that about 1.7A will pass through the MOSFET, well within its capabilities. The MOSFET is capable of dissipating up to 41W of power, and it will likely be consuming far less than that, so temperature should not be a concern. Finally, the question of switching speed. According to the data sheet, the combined time of the turn-on and turn-off delays is about 29ns, which if interpreted as the minimum switching time, gives a max frequency north of 34MHz. This analysis is not at all rigorous, but given that the delays are on the order of nanoseconds, and the maximum frequency of the PWM signal is in the kilohertz range (up to a max of about 20kHz), this part should be more than sufficient. The PWM frequency is also likely to be much lower, probably around or less than 1kHz.

Lastly there's the subject of the hotend return signal, which will utilize a thermistor external to this block in conjunction with a resistor divider in this block. This resistor is not a fixed value, as its value will depend on the nominal resistance of the thermistor, and therefore should be selected once a thermistor has been selected for the hotend. The important part of this setup is that it is ratiometric, relying on the logic level of the embedded device, which will come from that device's own integrated power supply. Therefore the reading returned to the device will not be as sensitive to supply voltage noise as it might otherwise be.

4.5.4 Interface Validation

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
embddd_dvc_mtrhtnd_cntrllr_dsig : Input		
Other: Motor/Hotend Control Signals	This interface provides control over the motors and hotend to the embedded device. It is a bundle of different signals, which would be overly tedious to represent as individual interfaces.	Informational.
Vmax: 5.25V	This is the maximum voltage expected out of the 5V DC power supply in the system, and therefore the maximum possible logic level that the control signals can be at.	The motor driver contained in this block has an IOREF pin that supports logic levels up to 5.5V, which means it could handle this potential maximum value. Additionally, the gate-source voltage of the transistor can be up to 20V.
Vmin: 0V	This is the minimum value that should be seen by this block on any digital signal sent from the embedded device, because this block will share a common ground with that device.	This block will share a common ground with the embedded device, therefore digital signals should never fall below ground. No special design considerations needed.
dc_t_dc_pwr_spply_mtrhtnd_cntrllr_dcpwr : Input		
Inominal: 2.9A	This value represents an estimated power draw from the system's 24V DC power supply of around 70W. This is based upon operating typical stepper motors at half-power, and the hotend heater at full power.	Under typical operation, the hotend heater will only need to be at full power to initially heat it, and then power can be reduced to maintain the temp. Additionally, not every motor is going to operated at the same time, so all coils at half power captures the typical power usage well enough. Thus during normal operation, it can be expected that the current should be around this value, and the block can handle far more as well due to the ratings on the drivers and the MOSFET.

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Ipeak: 9.8A	This value represents the maximum current that the 24V power DC power supply can give to this block, while still being able to power the 5V DC power supply to its own peak current.	Under normal operation, this block should never consume this much power, even though the supply is able to handle it. With the hotend and motors all running at max power the expected draw should only be about 4.1A, but even with a safe estimate of 5A for any power wiring there is still 4.8A left over. Therefore this block should never push the power supply to its limit.
Vmax: 25.2V	This is the maximum expected voltage of the 24V power supply (5% error).	The motor drivers and MOSFET both support up to 30V, so there is plenty of room for error here, and thus the power supply should not cause any problems for this block if its voltage is not entirely accurate.
Vmin: 22.8V	This is the minimum expected voltage of the 24V power supply (5% error).	The motor drivers can operate all the way down to 6V, so there is little concern there. The MOSFET doesn't technically have a lower limit (though current would suffer at very low voltages) and its performance will definitely not be affected at this minimum. While severely low voltages may stop the hotend heater core from generating sufficient heat, at 22.8V it should perform fine.

htnd_mtrhtnd_cntrllr_asig : Input

Other: Temperature Return Signal	In order to effectively control a heating element accurately, temperature feedback is needed.	Informational.
Vmax: 5.25V	This is the maximum voltage expected out of the 5V DC power supply in the system, and therefore the maximum logic level. Because this block supplies power from the embedded device's logic level to the thermistor used to take temp readings, this is the maximum value that can be seen on the return signal.	As this max value is external to this block, it only needs to be supported. This block must not use resistors that are low values for the resistor divider to avoid high currents. Using a resistor value of 1kOhm or larger will ensure that currents are never higher than about 5mA, even when the thermistor is effectively shorted.

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Vrange: 0-5V	This is the typical range of voltages for the return signal, based on the resistor divider setup. If the thermistor is effectively shorted, the output will be close to 0V. If the thermistor value is very large, the output will be close to the logic level, which is likely to be 5V.	As the thermistor occupies the portion of the divider attached to ground, this value can at absolute minimum be 0V, if the thermistor is shorted. It cannot fall below ground. If the thermistor is an open circuit, this value cannot go above the logic level voltage (up to Vmax at most).

mtrhtnd_cntrllr_embddd_dvc_dsig : Output

Other: Motor/Hotend Feedback	This interface provides feedback from the motors (SLA and errors) and hotend (temperature). It is a bundle of different signals, which would be overly tedious to represent as individual interfaces.	Informational.
Vmax: 5.25V	This is the maximum voltage expected out of the 5V DC power supply in the system, and therefore the maximum possible logic level that the control signals can be at.	The motor driver IOREF will limit the voltage of these return signals based on the logic level of the embedded device. The hotend return signal is also limited by this voltage, as described previously. There is a notable exception in the form of the SLA signal, which always has a range of 0V to 5V, regardless of IOREF, but this is still within the acceptable max range for this block.
Vmin: 0V	This is the minimum value that should be seen by the embedded device on any signal sent from this block, because this block will share a common ground with that device.	This block will share a common ground with the embedded device, therefore digital signals should never fall below ground. No special design considerations needed.

mtrhtnd_cntrllr_mtrs_asig : Output

Max Frequency: 20kHz	This is the maximum frequency at which the motors are likely to be operated. With micro-stepping, this will slow the motor operation down considerably, but speed is not a focus of this initial design.	According to the motor driver datasheet, the device internally uses PWM to control H-Bridge Drivers, and these PWM lines operate at a minimum of 20.5kHz, which is sufficient for this requirement.

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Other: 4-Wire Motor Drive Signal	This interface consists of a 4-wire connection for each motor, which there will likely be six of in the final design. This is to drive the two coils in the stepper motors.	Informational.
Vmax: 25.2V	This is the maximum expected voltage of the 24V DC power supply in the system, and therefore the maximum voltage the motor drivers can possibly present at their outputs.	As the motor driver is current-control focused, and due to how low resistance most stepper motor coils are, it's unlikely that this voltage will ever be present on the outputs of the drivers. This represents only a worst case potential, actual voltages will be much lower, around 2-5V across the positive and negative wires for a given coil, depending on the motor.
Vrange: 0-24V	This is based on the 24V DC power supply. As before, this is essentially the worst case range of voltages the driver can present on its outputs. Actual voltages will be lower.	Again, as stated before, the actual range depends on the motors selected, and will likely be closer to 0-5V different between two sides of a coil. The drivers will not present too much voltage for a given motor to handle, as they are current-control focused.

mtrhtnd_cntrllr_htnd_dsig : Output

Max Frequency: 1kHz	Represents the max frequency for the PWM signal that drives the hotend, based on the PWM signal the MOSFET recieves at its gate. This is not limited by the switching speed of the MOSFET, but instead the PWM frequency of the embedded device.	The MOSFET is capable of switching speeds in the MHz range, so it will have no trouble with switching at any speed up to and even beyond 1kHz, so long as it is supplied such a frequency by the embedded device. The device itself can support much higher frequencies, but this would be difficult to test without a signal generator given the limited PWM capabilities of most microcontrollers.
Other: Hotend Drive Signal	This signal is a PWM signal used to drive the hotend. Because generating smooth analog voltages is not an easy task, a PWM setup using a MOSFET is used.	Informational.

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Vmax: 25.2V	This is the maximum expected voltage of the 24V DC power supply in the system, and therefore the maximum voltage the MOSFET can output to the hotend.	As mentioned previously, the MOSFET can handle up to 30V, and therefore there is not a limiting factor for this voltage. There is no voltage gain involved in the hotend supply circuit, so this supply voltage cannot be surpassed.
Vmin: 0V	This is the voltage across the output when the MOSFET is off and not allowing any current to flow. As no current can flow, both sides of the output become the same voltage with respect to ground, and 0V with respect to each other.	This is the state of the output when the MOSFET is off, and the system cannot possibly go below this into reverse voltages, as the source of the MOSFET is connected directly to ground.

Table 4: Motor/Hotend Controller Interface Validation Table

4.5.5 Verification Plan

1. The block will first be powered with 25.2V, and VDD pin on the motor driver will be measured to ensure that it is in the specified range for the device.
2. The above step will be repeated for a supply voltage of 22.8V.
3. Next, a microcontroller will be connected to the block (loaded with a testing program) and the block will be powered from a supply that's about 24V.
4. A resistor or electronic load will be connected in place of the hotend heater core, and set up to measure current.
5. The testing program will use the MOSFET to output a PWM signal, which can then be measured by an oscilloscope. Current will also be measured here.
6. A set of two test resistors will be placed in for the resistor divider and thermistor, and the output will be read by a multimeter and compared to the logic level voltage from the microcontroller.
7. An electronic or other simulated load will be attached to the motor driver outputs (to prevent open or shorted coil faults).
8. The block will be set up to have motor driver and power supply currents measured.
9. The microcontroller will then instruct the motor drivers to step, and the current on the load shall be measured. (This may not be properly functional without a real motor to test with).
10. From the currents measured on the heating element and the motor driver, a max current for the block will be calculated and compared to the Ipeak and Inomimal values from the power supply interface. (As this test only involves one motor driver, that reading will need to be multiplied by 6).
11. If the measured values are all within acceptable ranges for given interfaces, this block can be considered verified.

4.5.6 References and File Links

4.5.6.1 References

4.5.6.2 File Links

- Motor Driver Carrier Board Info
- Motor Driver Carrier Board Schematic
- Motor Driver Datasheet
- MOSFET Datasheet

4.5.7 Revision Table

Date	Revision Info
1/7/2022	Sean Booth: Initial block validation draft.
1/21/2022	Sean Booth: Added schematic and design notes. Fleshed out general validation fully, as well as interface validations and verification plan.
3/6/2022	Sean Booth: Added to main project document.

4.6 DC to DC Power Supply

Champion: Sean Booth

4.6.1 Overview

This block supplies power to the embedded device in the system, and therefore performs a very essential function as it steps down the main DC supply voltage from around 24V to around 5V. This 5V supply will be connected to the AC-DC power supply and the embedded device via cables and connectors, and will have its own isolated PCB.

4.6.2 Design

At the black box level, this block takes one input and drives one output. It takes in power from the AC-DC power supply, and turns this 24V supply power into 5V supply power. The black box diagram can be seen in Figure 6 and the schematic for the block in Figure 7. The PCB design can be seen in Figure 8.

This design is centered around the TPS54232 DC-DC buck converter. Originally a different part was going to be chosen, but because the block champion had already created a working design with this central part, it was deemed simpler to just modify the design as necessary to fit the system requirements. The TPS chip supports up to 28V input voltage, but at higher voltages the efficiency does drop, even despite being a switching power regulator.

To best select components, the application notes for the TPS chip recommend using Texas Instruments' WEBENCH Power Designer, which aided in selecting compensation and biasing component values. From these values, suitable SMD and through-hole components were selected by availability on DigiKey. Capacitors are almost all Samsung Electro-Mechanics devices, and Resistors are from Stackpole Electronics. These were selected to exceed the voltage and power dissipation requirements of the supply. An indicator LED is included as an easy way to display that the system is functioning, at minimal current cost.

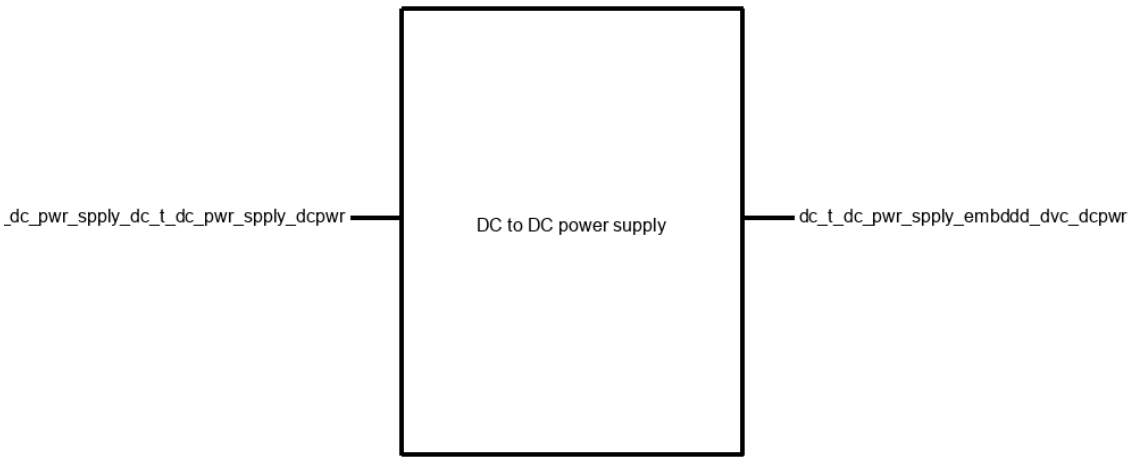


Figure 6: DC-DC Power Supply Black Box Diagram

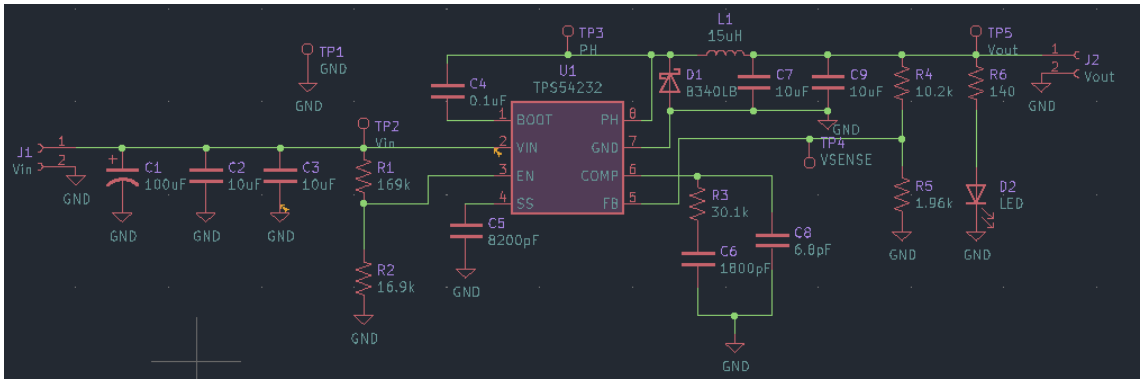


Figure 7: DC-DC Power Supply Schematic

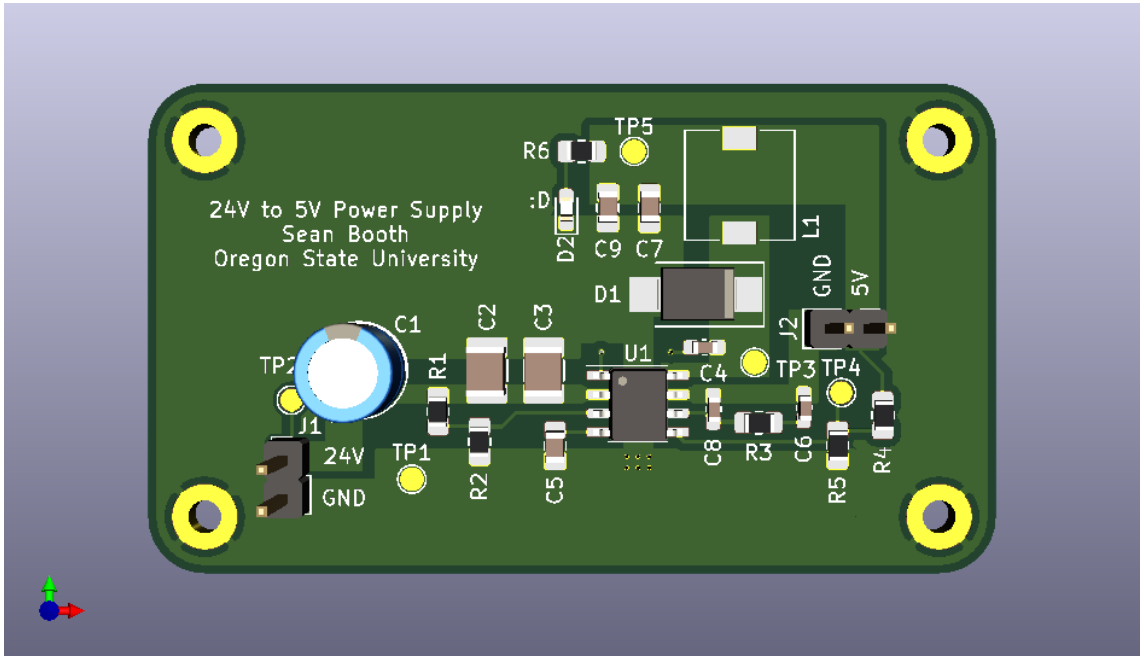


Figure 8: DC-DC Power Supply PCB

4.6.3 General Validation

The predecessor to this design has been tested with a 12V power supply, all the way up to around 18V, however the device can handle up to a 28V supply voltage, as per the TPS54232 datasheet. At this high input voltage efficiency will be lowered, but not to any degree that will matter given the low expected current draw from the embedded device. The minimum output voltage at 24V climbs to around 3V, which is below the 4.75V minimum for the output of this block.

The WEBENCH software was provided with the input and output parameters of the design, as well as the TPS chip that was going to be used, and therefore confidence in the viability of the design is high. An extra filtering capacitor is even included on the input to help with any noise coming from the 24V power supply. The feedback resistors should give an output voltage of about 4.96 V, which is within the acceptable 5% error bound. The catch diode is rated for 40V 3A, and thus will be more than sufficient for the design.

4.6.4 Interface Validation

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
ac_t_dc_pwr_spply_dc_t_dc_pwr_spply_dcpwr : Input		
Inominal: 125mA	This value is a reflection of the 5V nominal current from below, at about 80% efficiency.	The TPS chip can handle sinking 6A of current, as stated below.
Ipeak: 250mA	This value represents the maximum current that the 24V AC-DC power supply can give to this block, while still being able to power the motor/hotend controller to its own peak current. It also reflects the 5V peak current at around 80% efficiency.	The TPS chip is capable of sinking up to 6A of current, and PCB trace widths are suitable for 3A or more.
Vmax: 25.2V	This is the maximum expected voltage of the 24V power supply (5% error).	The TPS chip can handle supply voltages up to 28V according to its datasheet. This schematic also includes filtering for this supplied voltage. All components exposed to the input voltage are chosen to exceed this rated voltage by a good margin (usually 40-50V).
Vmin: 22.8V	This is the minimum expected voltage of the 24V power supply (5% error).	At a supply of 22.8V the TPS chip is able to supply up to 20V on the output. This is well above the 4.75V minimum the design targets.
dc_t_dc_pwr_spply_embddd_dvc_dcpwr : Output		
Inominal: 500mA	This value represents an estimated power draw from the system's 5V DC power supply of around 2.5W. This is a very generous estimate, and it's unlikely that the embedded device will ever come close to this value, let alone the peak current. This is more a measure of the 5V supply's capabilities.	The TPS chip is rated to provide 2A of current continuously, and the traces on the PCB are more than sufficient to provide this current. The STM32 board does not draw nearly this much current (less than 100mA in most cases).

Interface Property	Why is this interface this value?	Why do you know that your design details for this block above meet or exceed each property?
Ipeak: 1A	This value represents the maximum current that the 5V DC power supply can give to the embedded device (and anything connected to it) without disrupting the peak current the 24V supply has reserved for motor/hotend operation.	The TPS chip is capable of sourcing up to 6A of current, though the traces on the board are suitable for close to 3A, which is still well above this value.
Vmax: 5.25V	This is the maximum expected voltage of the 5V power supply (5% error).	The compensation components should put the output voltage around 4.96V, which is above this value. These components were selected by WEBENCH, which generates an application specific selection of components.
Vmin: 4.75V	This is the minimum expected voltage of the 5V power supply (5% error).	With the 24V supply voltage, the TPS chip can supply up to about 20V on the output. The compensation components should put the output voltage around 4.96V, which is above this value.

Table 5: DC-DC Power Supply Interface Validation Table

4.6.5 Verification Plan

1. Supply the block with 22.8V from a bench power supply.
2. Check that the output voltage is within the range of 4.75V to 5.25V.
3. Attach the 5V supply to an electronic load, and attempt to draw 1A of current from the device.
4. Monitor 5V supply output voltage and 24V supply output current during this test to ensure they are within the correct range.
5. Repeat the above steps with a supply voltage of 25.2V.
6. If the device meets the parameters during the two tests, it has been verified.

4.6.6 References and File Links

4.6.6.1 References

4.6.6.2 File Links

- TPS54232 Datasheet
- STM32L476RG Datasheet
- Resistor Datasheet
- Ceramic Capacitor Datasheet
- Electrolytic Capacitor Datasheet
- Inductor Datasheet
- Diode Datasheet
- LED Datasheet

4.6.7 Revision Table

Date	Revision Info
2/4/2022	Sean Booth: Initial block validation draft.
2/18/2022	Sean Booth: Revised schematic design and design section, added PCB design image. Added datasheets for components on the board. Minor revision to verification plan. Expanded general verification and filled out interface verification.
3/6/2022	Sean Booth: Added to main project document.

4.7 Control Board Block

Champion: Garrett Hallquist

4.7.1 Block Overview

The Control Board is the physical PCB that connects the Embedded Device and the Motor/Hotend Controller. It provides the contained blocks with the physical interface connections they need. The Control Board will be mounted on the mechanical design. This block is championed by Garrett Hallquist.

4.7.2 Block Design

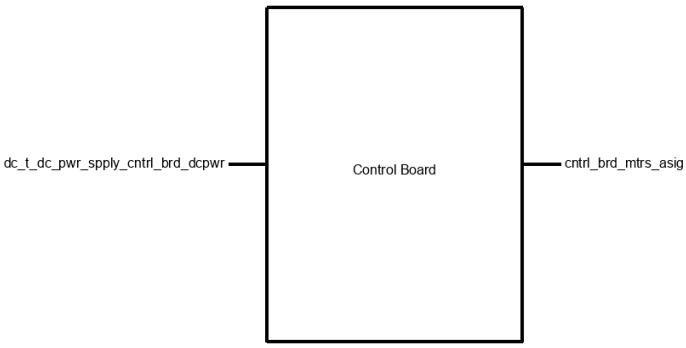


Figure 9: Control Board Black Box Diagram

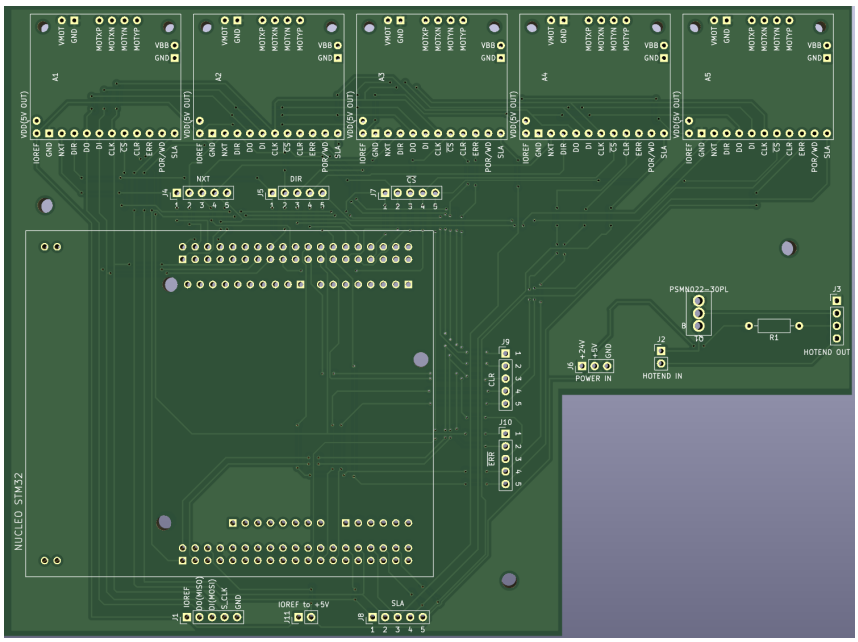


Figure 10: 3D Model of Control Board PCB

4.7.3 Block General Validation

The Control Board block is an important block for the electrical system of the Mobile 3D printer because it where the main control blocks, the Embedded Controller Block and the Motor/Hotend Controller Block, will be integrated. Connecting these blocks through PCB will increase signal accuracy, signal reliability, and circuit life time compared to a wired alternative.

Motor connections were linked to both the main controller and a set of pin holes. This allows for flexibility if a signal needs to be sent somewhere other than the original pin on the main controller. Data connections were routed with 12mil traces, and the 24V supply was routed with 80mil traces.These trace widths will carry the signals needed by through the board while being space efficient.

4.7.4 Block Interface Validation

PCB Requirements:

- The PCB must be less that 5.1in wide and 8.1in long.
- The PCB must be able to hold the main controller and the motor drivers.
- The PCB must be able to at least transmit the 5V and 24V power signals.

4.7.5 Block Testing Process

1. Show that the main controller and driver boards can be mounted on the PCB.
2. Show that the PCB is about 5in by 8in.
3. Connect a DC power source to the DC power input.
4. Measure 5V and 24V DC power Voltage at the motor/hotend dc power outputs to show that the traces work.

4.7.6 References and File Links

4.7.6.1 References

4.7.6.2 File Links

4.7.7 Revision Table

Date	Revision Info
2/4/2022	Garrett Hallquist: Added Initial Draft contents to all sections.
2/18/2022	Garrett Hallquist: Added design specific information that was not present in the first draft.
5/6/2022	Garrett Hallquist: Fixed formatting in design document.

4.8 Embedded Block

Champion: Garrett Hallquist

4.8.1 Block Overview

The Embedded Device is the microcontroller set-up that will contain all the 3D printer firmware, including the Inverse Kinematics Algorithms, Communication Code, and GCode Interpreter. It will accept printer instructions from the external connection, DC power from the DC to DC power converter, and feedback from the Motor/Hotend controller. It will output digital instructions to the Motor/Hotend Controller.

4.8.2 Block Design

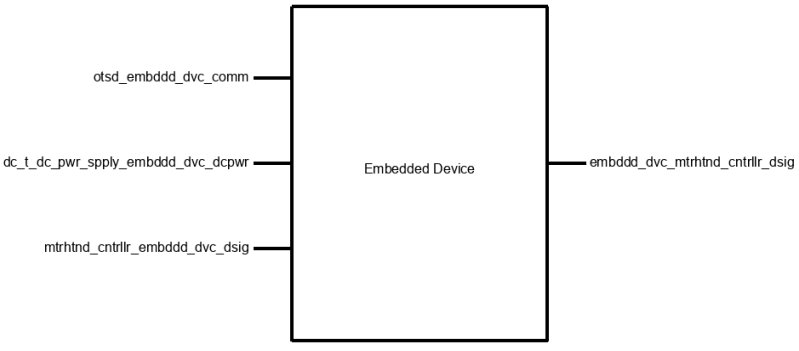


Figure 11: Embedded Black Box Diagram

4.8.3 Block General Validation

The embedded controller block is an integral part of the the system because is will utilize the firmware blocks to do all of the calculations for converting the Gcode input into a realistic tool path for the motor drivers. It is a central block which is needed for every block in the system to operate correctly, other than the power supplies.

We chose to use a STM32 Nucleo-64 board due to it's immediate availability at no cost, thanks to it being provided through the course. This allows us to start working with the board sooner, and potentially save budget for other parts. Additionally, if we find that we want a more powerful board, we can still upgrade if needed. The STM32 Nucleo-64 board meets this blocks needs by having a USB serial connection, 64 I/O pins and 1Mb of flash memory.

4.8.4 Block Interface Validation

Table 6: Serial Input otsd_embddd_dvc_comm

Interface Property	Why is this interface this value?	How do the design details meet expectations?
Datarate: 9600 baud	9600 baud is the standard serial communication baud rate.	From STM32 Nucleo-64 User Manual[1]: Supported USB Interfaces • Virtual COM Port (UM Features) • Mass Storage (UM Features) • Debug Port (UM Features)
Protocol: USB	USB is an easily accessible serial connection type, and can be used to connect directly to the Gcode source.	From STM32 Nucleo-64 User Manual[1]: • CN1 ST-Link USB mini B connector (UM Figure 3. Top Layout)
Other: Gcode Input	Gcode is the standard 3D printer/CNC tool path language.	The controller contains the firmware block which will interpret the Gcode.

Table 8: DC Power Input dc_t_dc_pwr_spply_embddd_dvc_dcpwr

Interface Property	Why is this interface this value?	How do the design details meet expectations?
V_{max}: 12V	The microcontroller recommends 7-12V input when drawing power from an external source.	From STM32 Nucleo-64 User Manual[1]: • V_{IN} from External Source = 7V to 12V (Table 7 in UM)
V_{min}: 7V	the microcontroller recommends 7-12V input when drawing power from an external source.	From STM32 Nucleo-64 User Manual[1]: • V_{IN} from External Source = 7V to 12V (Table 7 in UM)
I_{peak}: 800mA	The current should not go above this value to protect the parts in the embedded block.	From STM32 Nucleo-64 User Manual[1]: • Max Current from External Source (for $V_{IN} = 7V$) = 800mA. (Table 7 in UM) • Max Current from External Source (for $7V < V_{IN} <= 9V$) = 450mA. (Table 7 in UM) • Max Current from External Source (for $9V < V_{IN} <= 12V$) = 250mA. (Table 7 in UM)
$I_{nominal}$: 300mA	This is the expected current draw of the embedded block.	From STM32 Nucleo-64 User Manual[1]: • Max Current from External Source (for $V_{IN} = 7V$) = 800mA. (Table 7 in UM) • Max Current from External Source (for $7V < V_{IN} <= 9V$) = 450mA. (Table 7 in UM) • Max Current from External Source (for $9V < V_{IN} <= 12V$) = 250mA. (Table 7 in UM)

Table 10: Digital Signal Input mtrhtnd_cntrllr_embddd_dvc_dsig

Interface Property	Why is this interface this value?	How do the design details meet expectations?
V_{max}: 4V	This is the expected HIGH output signal strength of the encoder return signal.	For the STM32 in the LQFP64 Package[2]: • Input Voltage on Five-volt tolerant pins: $Max = V_{DD} + 4V$ (STM32 Electrical Characteristics) • Input Voltage on any other pins: Max = 4V (STM32 Electrical Characteristics)
V_{min}: 0V	This is the expected LOW output signal strength of the encoder return signal.	For the STM32 in the LQFP64 Package[2]: • Input Voltage on Five-volt tolerant pins: $Min = V_{SS} - 0.3V$ (STM32 Electrical Characteristics) • Input Voltage on any other pins: $Min = V_{SS} - 0.3V$ (STM32 Electrical Characteristics)
Other: Motor/Hotend Feedback	This is information sent from the motor/hotend encoders regarding data such as motor rotation and stall detection to be processed by firmware blocks.	The STM32 Nucleo-64 board has 16 digital I/O pins which can accept the digital signals.[1]

Table 12: Digital Signal Ouput embddd_dvc_mtrhtnd_cntrllr_dsig

Interface Property	Why is this interface this value?	How do the design details meet expectations?
V_{max}: 3.6V	This is the expected HIGH output signal strength of the controller output signal.	For the STM32 in the LQFP64 Package[2]: • Standard Operating Voltage $MaxV_{DD} = 3.6$ (STM32 Datasheet Table 14) • Standard Operating Voltage $MinV_{DD} = 1.8$ (STM32 Datasheet Table 14)
V_{min}: 1.8V	This is the expected LOW output signal strength of the controller output signal.	For the STM32 in the LQFP64 Package[2]: • Standard Operating Voltage $MaxV_{DD} = 3.6$ (STM32 Datasheet Table 14) • Standard Operating Voltage $MinV_{DD} = 1.8$ (STM32 Datasheet Table 14)
Other: Motor/Hotend Control Signals	This is information sent to the motor/hotend encoders is calculated by the firmware control blocks, including control instructions for the motors and hotend.	The STM32 Nucleo-64 board has 16 digital I/O pins which can output digital signals.[1]

4.8.5 Block Testing Process

1. connect STM32 Nucleo-64 Board to Computer through a serial USB.
2. To test that the block can receive power (testing dc_t_dc_pwr_spply_embddd_dvc_dcpwr), show that any signal can be read from a V_{DD} output port powered.
3. Use the USB connection to program the STM32 with a Validation Program.
4. connect a digital input wire and a digital output wire to two digital I/O pins.
5. Use the Validation Program to output a digital signal with to the output pin (testing embddd_dvc_mtrhtnd_cntrllr_dsig), then read the response in the input pin (testing mtrhtnd_cntrllr_embddd_dvc_dsig)
6. Show that the input was received by the controller through the USB connection.

4.8.6 References and File Links

4.8.6.1 References

[1] *Stm32 nucleo-64 boards (mb1136)*, UM1724, STMicroelectronics, 2020. [Online]. Available: https://www.st.com/resource/en/user_manual/um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf.

[2] *Arm®-based 32-bit mcu, 150 dmips, up to 1 mb flash/128+4kb ram, usb otg hs/fs, ethernet, 17 tims, 3 adcs, 15 comm. interfaces and camera"*, STM32F207xx, STMicroelectronics, 2020. [Online]. Available: https://www.st.com/resource/en/datasheet/cd00237391.pdf.

4.8.6.2 File Links

4.8.7 Revision Table

Date	Revision Info
1/7/2022	Garrett Hallquist: Added initial rough draft to each section.
1/21/2022	Garrett Hallquist: Replaced "TO BE ADDED" sections in Block Interface Validation with updated information. Improved Block Testing Process steps.
5/6/2022	Garrett Hallquist: Fixed formatting in design document.

5 System Verification Evidence

5.1 Universal Constraints

The image found in Figure 12 shows the system in an assembled state, with the cover off of the low-voltage enclosure to show the connectors and wiring. This image can be used to demonstrate the majority of the univeral constraints below, with the exception of the custom application and power supply efficiency, which provide further details in their sections.

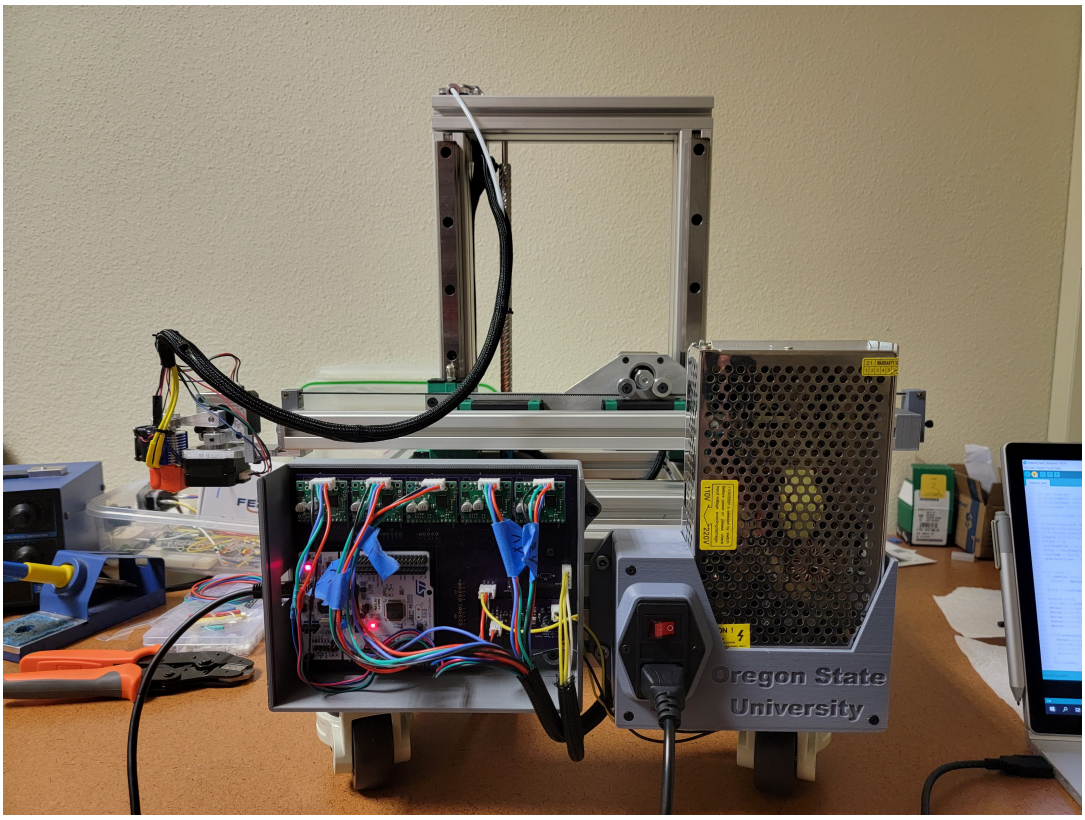


Figure 12: Image of the system, with the cover removed to show connectors.

5.1.1 Does not include breadboard

The system does not contain a breadboard, so this constraint is met. All connections are made via connectors, either screw-type or JST-XH connectors. All other connections are soldered. Daughter boards are connected to the main PCB via male/female header pairs.

5.1.2 Custom PCB and Application

The system consists of two student-designed PCBs which will be hooked up to the printer frame designed by the ME team. The printer will be driven by a custom program via USB connection. Both the main loop for the program and the PCBs have already been designed. The code for the project may be seen on the project GitHub, a link to which may be found in the file links for this section.

5.1.3 Rugged Enclosure

The electronics for the system have been enclosed in two enclosures mounted to the side of the printer body. The low-voltage and high-voltage electronics are separate, and the high-voltage wiring is entirely enclosed to reduce any risk of injury.

5.1.4 Wire connectors

All connections from PCB to PCB and from the PCBs to the motors will be through connectors and screw terminals.

5.1.5 Power Supply Efficiency

Testing of the power supplies in the system revealed that the DC-DC power supply (student design) was 79.7% efficient at moderate loads (0.5 A), and 84.3% efficient at max load (1 A). The AC-DC power supply was 70.9% efficient at moderate loads (3 A) and 77.8% efficient at max load (10 A). The efficiency for the AC-DC supply was estimated based off of measured apparent power and an estimated 0.75 power factor, though the actual power factor was likely worse than this. Still, even if the power factor were slightly higher, which is unlikely, the power supply would remain above 65% efficiency. The following table, Table 14, shows the measured data taken using an electronic load and a True RMS DMM.

Parameter	DC-DC Supply 0.5 A	1 A	AC-DC Supply 3 A	10 A
V_{in} (V)	25.2	25.2	126	126
I_{in} (I)	0.123	0.236	1.08	3.24
P_{in} (W)	3.10	5.86	136	408
V_{out} (V)	4.94	4.94	24.0	23.8
I_{out} (I)	0.500	1.00	3.01	10.0
P_{out} (W)	2.47	4.94	72.2	238
Power Factor	N/A	N/A	0.75	0.75
True P_{in} (W)	N/A	N/A	102	306
Efficiency	79.7%	84.3%	70.9%	77.8%

Table 14: Power supply efficiency data.

5.1.6 Less than 50% purchased modules

The only purchased modules in the system are the pre-made motor drivers and the microcontroller, which are mounted on the main control PCB, and the AC-DC power supply. All other components are student-made.

5.2 Positional Accuracy

5.2.1 Requirement

The system will move any motor in the system accurately to within $\pm 1^\circ$ of the target position.

5.2.2 Testing Process

1. The system will be instructed to move to a well-defined position within its range of motion.
2. The system will then attempt to move to this position.
3. The difference between the actual position and the targeted position will be measured.
4. If the system is always within the threshold, it passes this requirement.

5.2.3 Testing Evidence

To measure the exact rotation of a motor the XY axial motor is aligned with a protractor. In debug mode, the motor is commanded to rotate 45°. The difference between the the initial value and the end value was calculates to be 45°±1°. A video showing this is linked below.

- Arbitrary Positioning and Positional Accuracy Video

5.3 Five Motor Control

5.3.1 Requirement

The system will control 5 motors at once.

5.3.2 Testing Process

1. The system will be instructed to move all the motors at the same time.
2. All 5 of the motors must move during this test, approximately the same amount.
3. If the system can move all 5 motors simultaneously, it passes this requirement.

5.3.3 Testing Evidence

Motors were operated by the microcontroller via the motor drivers on the control board, all powered by the AC-DC and DC-DC power supplies. All 5 motors were capable of moving simultaneously. A video showing this is linked below.

- Five Motor Control Video

5.4 Filament Extrusion

5.4.1 Requirement

The system will extrude filament to an accuracy of $\pm 5 \frac{\text{mm}}{\text{m}}$.

5.4.2 Testing Process

1. The system will be instructed to extrude a controlled amount of filament.
2. The system will then attempt to extrude this amount of filament.
3. The difference between the target and actual extrusion length will be measured.
4. If the system is always within the threshold, it passes this requirement.

5.4.3 Testing Evidence

The extruder was commanded to extrude 40cm of filament. After all of the filament was extruded, the length of the extruded filament was measured with a ruler and found to be within ±1mm of 40cm, which is within the acceptable range. A video showing this is linked below.

- Filament Extrusion Video

5.5 Software Self Collision Avoidance

5.5.1 Requirement

The system's software will prevent motors from moving to angles which would cause a self-collision or otherwise be impossible due to the mechanical frame. This is not meant as a replacement to hardware collision avoidance, but as an extra layer of safety.

5.5.2 Testing Process

1. The system will be instructed to move in a way that would put a motor angle out of bounds.
2. The system's software will then abort the print job.

5.5.3 Testing Evidence

A helper function has been added to the program's kinematics.c file in order to handle changing motor angles during kinematics (see references and file links section for a link). The function will return one if the angle is not withing the motor's bounds. The inverse kinematics function in the same file is checking for this, and will also return one in that case. The main function is checking for the return value of the inverse kinematics function, and will break in that case. A video showing this is linked below.

- Software Self Collision Avoidance Video

5.6 Arbitrary Positioning

5.6.1 Requirement

The system will rotate motors independently to at least 20 degrees from its starting position

Verification process:

1. The system will be set to a known position.
2. The system will be instructed to move to a point where all motors are more than 20 degrees from their original position
3. The system will then attempt to reach this point.
4. If the system is able to move the motors, it passes the requirement.

5.6.2 Testing Evidence

To measure the exact rotation of a motor the XY axial motor is aligned with a protractor. In debug mode, the motor is commanded to rotate 45°. The difference between the the initial value and the end value was calculates to be 45°±1°, which is greater than 20°, meeting this requirement. A video showing this is linked below.

- Arbitrary Positioning and Positional Accuracy Video

5.7 Collision Detection

5.7.1 Requirement

The system will detect if it has collided with an object or itself and stop all motors.

5.7.2 Testing Process

1. The system will be instructed to move to some position.
2. The system will then be physically obstructed so that it collides with a safe object.
3. The system should then stop all movement when it detects this collision.
4. If the system stops every time it collides with an object, it passes this requirement.

5.7.3 Testing Evidence

The YZ axial motor was commanded to rotate in a way that would cause the printer to collide with an object. When the printer hit the object and the motor started to stall, the motor was commanded to stop moving and the program returned a stall detection error. A video showing this is linked below.

- Collision Detection Video

5.8 Instruction Interface

5.8.1 Requirement

The system will accept instructions through a wired serial interface, and will acknowledge the receipt of instructions. Invalid instructions are to be given an error message in reply.

5.8.2 Testing Process

1. The system will be given some valid or invalid instruction.
2. The system will then acknowledge the instruction, and give an error if it is invalid.
3. If the system responds correctly to the instruction, it passes this requirement.

5.8.3 Testing Evidence

In debug mode, a command is sent from a computer to the printer through a serial connection that would cause the motors to move. After the command is sent, the motors move and the computer gets a response message from the printer. A video showing this is linked below.

- Instruction Interface Video

5.9 Thermal Runaway Protection

5.9.1 Requirement

The system will attempt to detect and stop any instances of thermal runaway. If expected and real temperature do not match for 10 s, the system will shut off all heating elements and require user intervention before it can start again.

5.9.2 Testing Process

1. The temperature sensor will be removed from the influence of the heating element.
2. The system will be instructed to heat up the hotend to 200 °C
3. The system should shut off the heating element within 20 s when it detects the fault.
4. If the system shuts down the heating element properly, it passes this requirement.

5.9.3 Testing Evidence

The system was tested with multiple failure conditions, and was able to correctly detect the fault in both cases. A video showing two of these tests can be found linked below, specifically it shows the system responding to the sensor being decoupled from the heater core, and to the sensor being disconnected entirely. The system will also shut down the hotend in the case that the temperature climbs too high above the set point, or above a max overall temp.

- Thermal Runaway Protection Video

5.10 Safety

5.10.1 Requirement

The system will contain a physical shutoff switch that can be used to disconnect power from the system in case of unexpected behavior. The system will also enclose any high voltage electronics and warning labels will be placed where hazards (such as hot areas) are present.

5.10.2 Testing Process

1. The system will be powered on.
2. The shutoff switch will then be pressed, and the system should immediately lose all power.
3. The system will also be visually inspected for exposed high voltages, and unlabeled hazards.
4. If the system shuts off as expected and there are no unlabeled, exposed hazards, the system passes this requirement.

5.10.3 Testing Evidence

The printer was visually inspected for exposed high voltage nodes and high temperature locations. All these locations are encased and/or labeled. The printer also has an easily accessible off switch. When the switch is flipped, the motors visually lost power. A video showing this is linked below.

- Safety Video

5.11 References and File Links

5.11.1 References

5.11.2 File Links

- Project Github
- kinematics.c
- Software Self Collision Avoidance Video
- Five Motor Control Video
- Thermal Runaway Protection Video
- Arbitrary Positioning and Positional Accuracy Video
- Filament Extrusion Video
- Safety Video
- Instruction Interface Video
- Collision Detection Video

5.12 Revision Table

Date	Revision Info
3/6/2022	Preston Pickering: Created section header and laid out outline. Added explanations for the universal constraints I am qualified to talk about.
3/6/2022	Preston Pickering: Added section for self collision avoidance.
3/6/2022	Sean Booth: Updated five motor control requirement and universal constraints.
3/13/2022	Preston Pickering: Updated software self collision avoidance and added evidence and links to evidence.
3/14/2022	Sean Booth: Added video evidence link for five motor control.
4/22/2022	Sean Booth: Added other requirements to section, pending evidence and some requirements edits.
5/2/2022	Sean Booth: Edits to some testing procedures and requirements to reflect changes discussed. Added video for Thermal Runaway Protection. Updated universal constraints section.
5/2/2022	Garrett Hallquist: Added evidence for 7 requirements. Edited testing procedures to match evidence shown.
5/5/2022	Garrett Hallquist: edited evidence formatting to match instructor feedback.
5/6/2022	Sean Booth: Added evidence to the universal constraints as per feedback.

6 Project Closing

6.1 Future Recommendations

6.1.1 Technical Recommendations

1. 3D Printer Controller Board Recommendation: We spent a lot of time designing a custom printer board for this project. While we are proud of our design, there are many 3D printer controller boards that can be bought that are superior to our design in most aspects (for example, the SKR Pro motherboard from BIGTREETECH). If we had chosen to purchase a 3D printer board rather than design one, we could have worked on other aspects of the project and possibly increased the scope. Overall, we would recommend a future team to consult with the Capstone professors and project partner about purchasing any parts that already exist, so that more time can be spent pushing the project further.

2. Separation of Mechanical and Electrical Design Recommendation: When the project first started, the mechanical design team had not decided on a mechanical design. This made it difficult for the EECS team to start design work when we weren't sure what we were designing for yet. We waited until mechanical had a fairly consistent design before starting our work, leaving us with less time to work on the project in general. If we had been more proactive and talked with the mechanical team about what controls will be needed no matter the design, we could have begun the design much earlier and gotten more work done. We would recommend that a future team define the system's electrical input and output interfaces as soon as possible so that the electrical design can be completed independently of the mechanical design before integrating the two. Of course, doing this would require strong communication from both sides to prevent false assumptions and other misunderstandings.

3. Use Outside Libraries for Difficult Tasks: Due to confusion about the program component of the universal project requirements, we chose to develop our own Inverse Kinematics functions. This was a mistake, as Inverse Kinematics is a very difficult problem. When writing software it is generally preferable to solve the most abstract version of the problem you are working on. That way the same code can be re-used later. The general case for Inverse Kinematics has been solved already and is in use in robots the world over, but was beyond the skill of our programming team. We settled for a approach specific to the project. Still, coming to this realization took valuable time which could have been better used elsewhere. Using an external library would have been the way to go. This portion of the project is only now mostly working in simulation despite hours of work, and has not been tested on the printer itself. Even when using an external library, a lot of time would have been needed to calibrate it to the actual machine. Avoiding the temptation to unnecessarily write your own libraries

when there are available alternatives is essential, especially as tasks may be more difficult than anticipated.

4. Use Limit Switches and Encoders Where Relevant: One of the design decisions we made was to use stall detection as a method for homing the printer axes, but this proved to be problematic in the end as the feedback from the motor drivers was sometimes ambiguous, and the mechanical design also made it harder to use stall detection for homing on certain axes. One axis that posed a particular issue was the Z-axis motor, which had to be equipped with a 10-to-1 gearbox to allow it to drive the very heavy armature. This meant that it would essentially never stall, and would instead begin to pull the machine apart if run too far in one direction. We managed to work around this with a few tricks, but it would've been better to simply equip one end of each axis with a limit switch for homing on the prismatic joints, and equipped the rotational joints with absolute encoders so that homing them would not be necessary. At that point, stall detection would become a safety system and not integral to the operation of the printer. Absolute or incremental encoders could provide a lot of useful feedback into the system, though they come with both a monetary and complexity cost, thus it is recommended to use the absolute encoders only on the rotational axes [1].

6.1.2 Global Impact Recommendations

1. Safety and Health Impacts Recommendations: Future teams should always be aware possible accidents that could occur when working with a 3D printer [2]. Currently, the 3D printer uses stall detection to prevent crushing/trapping accidents, and the printer does not move fast enough yet to cause impact accidents. If the motors or motor drivers are changed in the future, we would recommend using limit switches, either along side or instead of stall detection because they can be more flexible to design around. The heated elements have been labels as such, and we recommend continuing the practice. For further safety, one suggestion is to use warning lights to show when the heated elements are activated.

2. Environmental Impacts Recommendations: Some of the environmental impact of plastic 3D printing can be alleviated by using biodegradable filament. By using biodegradable filament such as PLA, a cornstarch-based filament, the environmental impact of 3D printing would be reduced. Another way to reduce plastic waste is to use filaments made from recycled plastic [3]. By reusing old plastic waste, printing with the filament will not cause an increase in plastic waste.

6.1.3 Teamwork Recommendations

1. Meet In-Person: When meeting with the team, professor, or project partner we tried to meet in person if possible. In-person meetings make it easier to include everyone in a discussion and can just generally make group members more comfortable around each other [4]. We started the project with weekly team meetings in-person, but later on we started slipping had met face-to-face less and less. It became pretty apparent that the less we met in person, the less general communication we had and the less we knew about what the other group members were working on. Scheduling in person meetings was sometimes a struggle due to timing conflicts, but when we did meet the meetings were more productive and well work the effort. We would recommend future teams to set up weekly or even bi-weekly meetings in-person so make sure the group can communicate effectively.

2. Ask For Help: We have maintained strong communication within our group and with the project partner throughout the project through weekly meetings, but we did not ask for help enough. During our meetings, we had the mindset that we were working our section of the project so it was our responsibility. This lead to the meetings mostly being progress reports and future plans. While those are productive topics, it mostly meant we were keeping our struggles to ourselves. If we had gone to the project partner and specifically asked for help or advice in places we were struggling, we could have save a lot of time worrying about problems that we were not sure how to solve [5]. We would recommend actively asking the project partner and professors for help when you don't know what to do.

6.1.4 References

[1] J. Smoot. "Rotary encoder options: Absolute or incremental?" (2018), [Online]. Available: https://www.digikey.com/en/articles/rotary-encoder-options-absolute-or-incremental.

[2] NIOSH. "3d printing safety at work," The National Institute for Occupational Safety and Health. (2020), [Online]. Available: https://www.cdc.gov/niosh/newsroom/feature/3Dprinting.html.

[3] A. Team. "Eco-friendly 3d filament." (2021), [Online]. Available: https://www.aniwaa.com/guide/am-materials/eco-friendly-3d-filament/.

[4] I. E. Team. "9 reasons why face-to-face meetings are important." (2021), [Online]. Available: https://www.indeed.com/career-advice/career-development/face-to-face-meetings.

[5] J. Davis. "4 tips to effectively ask for help—and get a yes." (2020), [Online]. Available: https://www.psychologytoday.com/us/blog/tracking-wonder/202002/4-tips-effectively-ask-help-and-get-yes.

6.2 Project Artifact Summary with Links

We have produced a number of project artifacts. Most of these can be found on the github for the project, which contains the "production" C and Arduino code as well as several programs designed for testing essential components. The github page also contains circuit diagrams and a breakdown of our serial communication protocol. In addition to this we have created a number of videos showing off our engineering requirements.

6.2.1 Links

- Project Github
- 3D Printer Requirement Video Playlist

6.3 Presentation Materials

- EECS Group 30 Project Poster

6.4 Revision Table

Date	Revision Info
5/6/2022	Garrett Hallquist: Added Controller Board, MIME/EECS separation, In-Person, safety impacts, environment impacts, and Ask for Help recommendations to the Future recommendations section. Changed various latex formatting. Added References section.
5/6/2022	Preston Pickering: Added another technical recommendation (on using libraries for difficult components), added information about project artifacts and linked to the github.
5/6/2022	Sean Booth: Added technical recommendation regarding limit switches and encoders.