# Automated Front Panel Testing for Tektronix

By: Kevin Ho, Ryan Christensen,
Felipe Orrico Scognamiglio, Dennis Kichatov

**Oregon State University**
**ECE 44x: Senior Capstone**

# Legal Notice

# Table of Contents

**Legal Notice** **1**

**Table of Contents** **2**

**1. Overview** **8**

**1.1. Executive Summary** **8**

**1.2. Team Communications Protocols and Standards** **8**

**1.3. Gap Analysis** **10**

**1.4. Timeline/Proposed Timeline** **11**

**1.5. References and File links** **13**

1.5.1. References (IEEE) 13

1.5.2. File Links 13

**1.6. Revision Table** **13**

**2. Requirements Impact and Risks** **14**

**2.1. Requirements** **14**

2.1.1. Project Requirement 1: The system will press a button 14

2.1.2. Project Requirement 2: The system turns multiple sized knobs 14

2.1.3. Project Requirement 3: The system will have autonomous operation. 14

2.1.4. Project Requirement 4: The system will allow a user to manually control the device. 15

2.1.5. Project Requirement 5: The system will be able to toggle power to the stepper motors through the GUI. 15

2.1.6. Project Requirement 6: The system will produce a video feed 15

2.1.7. Project Requirement 7: Fits in a defined size 15

2.1.8. Project Requirement 8: The system will display commands on screen during video playback 16

**2.2. Design Impact Statement** **16**

2.2.1 Public Safety 16

2.2.1.1 Impacts 16

2.2.1.2 References IEEE 17

2.2.1.3 Revision Table 17

2.2.2 Welfare Impacts 17

2.2.2.1 Impacts 17

2.2.2.2 References IEEE 18

2.2.2.3 Revision Table 18

2.2.3 Cultural and Social Impacts 18

2.2.3.1 Impacts 18

2.2.3.2 References IEEE 18

2.2.3.3 Revision Table 18

2.2.4 Environmental Impacts 18

2.2.4.1 Impacts 18
2.2.4.2 References IEEE 19
2.2.4.3 Revision Table 20
2.2.5 Economic Impacts 20
2.2.5.1 Impacts 20
2.2.5.2 References IEEE 21
2.2.5.3 Revision table 21
**2.3. Risks 22**
**2.4. References and File Links 24**
2.4.1. References 24
2.4.2. File Links 24
**2.5. Revision Table 24**
**3. Top-Level Architecture 25**
**3.1. Block Diagram 25**
**3.2. Block Descriptions 26**
3.2.1 User Interface: 26
3.2.2 Power Supply: 27
3.2.3 Hardware Control API: 27
3.2.4 Motor Control: 28
3.2.5 Interrupts 29
3.2.6 Mechanical Structure: 29
3.2.7 Enclosure: 30
3.2.8 Stepper Motors: 30
**3.3. Interface Definitions 31**
3.3.1 User Interface 31
3.3.2 Hardware Control API 33
3.3.3 Mechanical Structure 35
3.3.4 Stepper Motors 36
3.3.5 Interrupts 37
3.3.6 Motor Control 38
3.3.7 Enclosure 41
3.3.8 Power Supply 41
**3.4. References and File Links 43**
3.4.1. References (IEEE) 43
3.4.2. File Links 43
**3.5. Revision Table 43**
**4. Block Validations 44**
**4.1. Mechanical Frame - Ryan Christensen 44**
4.1.1. Block Overview 44
4.1.2. Block Design 44

4.1.3. Block General Validation 44
4.1.4 Interface Validation 45
4.1.5. Block Testing Process 45
4.1.6. References and File Links 46
4.1.6.1. References (IEEE) 46
4.1.6.2. File Links 46
4.1.7. Revision Table 46
**4.2. Stepper Drivers - Ryan Christensen 46**
4.2.1. Block Overview 46
4.2.2. Block Design 46
4.2.3. Block General Validation 46
4.2.4 Interface Validation 47
4.2.5. Block Testing Process 48
4.2.6. References and File Links 49
4.2.6.1. References (IEEE) 49
4.2.6.2. File Links 49
4.2.7. Revision Table 49
**4.3. Interrupts - Kevin Ho 49**
4.3.1. Block Overview 49
4.3.2. Block Design 50
4.1.3. Block General Validation 51
4.3.4. Block Interface Validation 51
4.3.5. Block Testing Process 52
4.3.6. References and File Links 53
4.3.6.1. References (IEEE) 53
4.3.6.2. File Links 53
4.3.7. Revision Table 53
**4.4. Motor Control - Kevin Ho 53**
4.4.1. Block Overview 53
4.4.2. Block Design 54
4.4.3. Block General Validation 55
4.4.4. Block Interface Validation 55
4.4.5. Block Testing Process 58
4.4.6. References and File Links 59
4.4.6.1. References (IEEE) 59
4.4.6.2. File Links 59
4.4.7. Revision Table 59
**4.5. Power Supply - Dennis Kichatov 60**
4.5.1. Block Overview 60
4.5.2. Block Design 60
4.5.3. Block General Validation 63

4.5.4 Block Interface Validation 64
4.5.5. Block Testing Process 65
4.5.7. References and File Links 66
4.5.7.1. References (IEEE) 66
4.5.7.2. File Links 66
4.5.7. Revision Table 66
**4.6. Enclosure - Dennis Kichatov 67**
4.6.1. Block Overview 67
4.6.2. Block Design 67
4.6.3. Block General Validation 69
4.6.4 Block Interface Validation 70
4.6.5. Block Testing Process 70
4.6.6. References and File Links 70
4.6.7. Revision Table 70
**4.7. Hardware Control API - Felipe Orrico Scognamiglio 71**
4.7.1. Block Overview 71
4.7.2. Block Design 71
4.7.3. Block General Validation 73
4.7.4. Block Interface Validation 73
4.7.5. Block Testing Process 76
4.7.6. References and File Links 76
4.7.6.1. References (IEEE) 76
4.7.6.2. File Links 76
4.7.7. Revision Table 76
**4.8. Graphical User Interface - Felipe Orrico Scognamiglio 78**
4.8.1. Block Overview 78
4.8.2. Block Design 78
4.8.3. Block General Validation 80
4.8.4. Block Interface Validation 81
4.8.5. Block Testing Process 83
4.8.6. References and File Links 83
4.8.6.1. References (IEEE) 83
4.8.6.2. File Links 83
4.8.7. Revision Table 83
**5. System Verification Evidence 84**
**5.1. Universal Constraints 84**
5.1.1. The system may not include a breadboard 84
5.1.2. The final system must contain both of the following: a student designed PCB and a custom Android/PC/Cloud application 84
5.1.3. If an enclosure is present, the contents must be ruggedly enclosed/mounted 84

5.1.4. If present, all wire connections to PCBs and going through an enclosure (entering or leaving) must use a connector 84
5.1.5. All power supplies in the system must be at least 65% efficient 85
5.1.6. The system may be no more than 50% built from purchased modules 85
**5.2. The system will press a button. 85**
5.2.1. Requirement 85
5.2.2. Testing Process 85
5.2.3. Testing Evidence 85
**5.3. The system turns multiple sized knobs. 85**
5.3.1. Requirement 85
5.3.2. Testing Process 86
5.3.3. Testing Evidence 86
**5.4. The system will have autonomous operation. 86**
5.4.1. Requirement 86
5.4.2. Testing Process 86
5.4.3. Testing Evidence 86
**5.5. The system will allow a user to manually control the device. 86**
5.5.1. Requirement 86
5.5.2. Testing Process 86
5.5.3. Testing Evidence 87
**5.6. The system will be able to toggle power to the stepper motors through the GUI. 87**
5.6.1. Requirement 87
5.6.2. Testing Process 87
5.6.3. Testing Evidence 87
**5.7. The system will produce a video feed 87**
5.7.1. Requirement 87
5.7.2. Testing Process 87
5.7.3. Testing Evidence 87
**5.8. The Gantry System fits within a defined size 88**
5.8.1. Requirement 88
5.8.2. Testing Process 88
**5.9 The system will display commands on screen during video playback 88**
5.9.1. Requirement 88
5.9.2 Testing Process 88
5.9.3. Testing Evidence 88
**5.10. References and File Links 88**
5.10.1. References (IEEE) 88
5.10.2. File Links 89
**5.11. Revision Table 89**

**6. Project Closing 89**
**6.1. Future Recommendations 89**

6.1.1. Technical recommendations 89
6.1.2. Global impact Recommendations 90
6.1.3. Teamwork Recommendations 90
**6.2. Project Artifact Summary With Links 91**
6.2.1 PCB and Schematic Diagrams 91
6.2.2 Mechanical Structure Fusion360 Designs 91
6.2.3 GRBL HAL file for Black Pill 91
6.2.4 Enclosure Models 91
6.2.5 User Interface / Hardware Control API Source Code 91
6.3. Presentation Materials 91
**6.4. References and File Links 91**
6.4.1. References (IEEE) 91
6.4.2. File Links 91
**6.5. Revision Table 91**

**A. Appendix 92**

# 1. Overview

## 1.1. Executive Summary

To help Tektronix both test their devices as well as offer remote operations for engineers working at home, they have requested a device which can be used both in a laboratory setting and remotely accessed. Our team has designed a gantry-style oscilloscope validation robot for Tektronix. This robot is able to push buttons and turn the knobs of many different types of Tektronix oscilloscopes and aims to allow remote testing and validation. Similar to a 3D printer our robot will be able to autonomously test and run validation from scripts provided by Tektronix engineers. The device also allows manual operators to work with the device on-site. With this approach, we can program the robot to interface with any type of oscilloscope Tektronix may use, and whatever testing procedure they need. This document serves to give people a better understanding of the system at large and the technical aspects of it, with unique challenges across multiple disciplines, including mechanical, electrical and computer engineering, and computer science.

The team for the Automated Front Panel Testing consists of four individuals. Kevin Ho, Dennis Kichatov, Felipe Orrico Scognamiglio, and Ryan M Christensen. All of the members major in Electrical and Computer engineering and we are all very excited to tackle this project and all its challenges. Since this is a partner project, the team will be working closely with our partner, Tektronix. Tektronix is a company that specializes in creation and manufacturing of electrical testing equipment. During this project's lifetime our team of four will be working with a team of staff members from Tektronix, under Ancil Tucker, to assist along the way.

## 1.2. Team Communications Protocols and Standards

**Each member of the team can be contacted at their individual emails:**

Ryan Christensen: chrisrya@oregonstate.edu
- Point of contact

Kevin Ho: hoke@oregonstate.edu
Dennis Kichatov: kichatod@oregonstate.edu
Felipe Orrico Scognamiglio: orricosf@oregonstate.edu

**Project Partner contact information:**

Ancil Tucker: ancil.tucker@tektronix.com

| Topic | Protocol | Standard |
|---|---|---|
| Due dates for deliverables | The assigned work, either the block or design, must be completed before the assigned dead-line. This gives the team time to discuss and revise the work if needed. | The work is completed and the requirement that is set is fully filled. The format is based off of Canvas or set by the team. |
| Task management | The team will use Asuna to keep track of progress and tasks. | Each individual member is required to update their progress on Asuna every week. |
| Issue occurrence | When an issue that can't be solved occurs, contact the team through Discord to discuss the problem. If the problem is still not solved, the team will schedule a meeting time. | Every team member must be honest and open to listening to each other. Every member will try to help solve the problem. |
| Requirement List | Requirements for each block are set and written down on a shared Google Doc that all members have access to. | Each member is responsible to check and make sure their block meets the requirement set before the deadline. |
| Team Google Drive | All documents, designs and schematics are uploaded to the shared Google drive. The drive is also formatted with folders for organization. | Each member is responsible to upload their own work on the shared drive and into the correct folder. |
| Absence | Team members must give notice to other team members if they cannot make a meeting. | The team members have to be responsible to reach out and catch up with what happened during the meeting. |

Table 1: Team Standards

Project Partner Communication Analysis

- The purpose for this project is to automate the front panel testing and give employees a remote way to access oscilloscope testing. The project partners are to give advice, review the functionality, and provide additional knowledge to complete tasks.
- The project partners consist of a general project manager, and 4 other software/electrical engineers from Tektronix.
- The project partners would like to know progress on the project and any questions that the team has about the requirements.
- The project partners have not spent too much time on the implementation of the goal.
- The team will contact through email either to the project manager or directly to one of the Tektronix team members once or twice a week. The team can also schedule team video meetings done on microsoft teams.

Keeping the project partner and teammates informed is important to complete the task. To keep the project partner informed, the team will email the partner on Mondays and Fridays. Monday reports will go over tasks the team is working on throughout the week. Friday reports will tell the project partner what progress has been made within the tasks. Within the team, team members will update progress done on tasks in a group chat, discord. Team members must be honest with their progress. When there is an issue the team member will first message the team. If the issue can not be resolved through messages, the team will schedule a meeting to discuss the problem. Team members also must listen to each other to create an inclusive environment where everyone can ask questions and share ideas.

Blocks must have clear and concise requirements to avoid future confusion and error. These requirements are written down on a document that all team members have access to. This helps the project partner understand the function of the block. Having clear requirements helps team members know what the block should or should not do. Block requirements also have to be within the scope of the project. Requirements cannot be unrealistic for the project.

## 1.3. Gap Analysis

This project exists as a quality of life improvement to the Tektronix product validation and development team. It will allow Tektronix team members to manipulate the physical panels (buttons and knobs) of their oscilloscopes and to read data remotely. Tektronix is interested in this project because it is a way to have their engineers test products from home. During the current global pandemic, Tektronix employees either decided to work from home or had to work from home due to laws and regulations set out by the state. This exposed a gap within their current workflow that if engineers needed to work with hardware, they needed a way to access the oscilloscopes without being able to work in their labs. Tektronix also believes that our design could be used to validate and test the limit of their oscilloscopes. By having an automated robot doing repetitive testing, the engineers are free to work on more important tasks. For this design project, we were given freedom to choose whatever solutions we deem fit for the problem as long as we properly fulfill the requirements proposed by Tektronix. More information about project requirements and needs can be found in section 2 of this report.

# 1.4. Timeline/Proposed Timeline

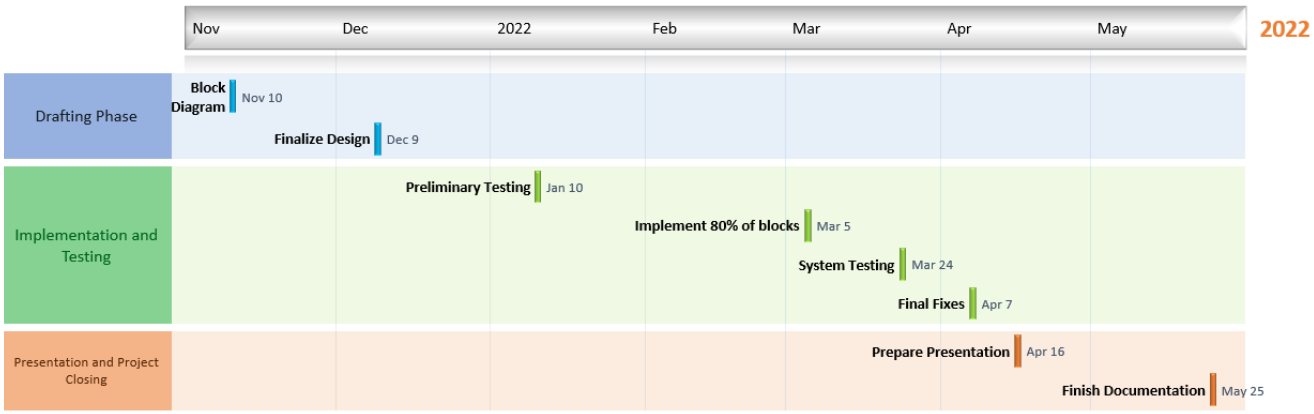



Figure 1: Timeline

The project will be divided into three main phases. Drafting Phase (current phase), Implementation and Testing phase, and Presentation and Project Closing phase. Each phase has a different length and some overlapping is expected as we reach the end of the project development and implementation.

The Drafting Phase is about sixty five days long (from October first to December fifth). During this phase, the project is in the drafting stage, meaning that it is focused on the development of ideas and methods.

The Implementation and Testing phase is about four and a half months long and focuses on the implementation and testing of the design created during the initial phase. During this time, production and assembly of all necessary components and code will take place. Changes to the design get increasingly more difficult to accomplish, and therefore must be avoided.

The Presentation and Project Closing phase focuses on the finalization of development and testing, the preparation of a presentation about the project, and the finalization of the project documentation. This phase is about one month long, and is the last phase of the project before it is delivered to Tektronix as a finished product.

| Goals | Current States | Future States | Gap(difference between states) | Challenges | Solutions |
|---|---|---|---|---|---|
| Meet the Tektronix team/partner | Introductions Made | Meeting on 10.26.21 | Not meet team yet | Virtual meeting on microsoft teams. | Wait till Tuesday 10.26.21 |
| Rough sketch of diagram | Not started yet | Block diagram for every aspect of project | No blocks have been sketched | Need to know parameters and function of project | Need to know parameters and function of project |
| Set a timeline | finished rough timeline | Completed timeline | Timeline is in the works | Need to know parameters and function of project | Need to know parameters and function of project |
| Research for project | Started | Understand what blocks are needed and how to implement them. | Not sure what blocks we need. | Need to know parameters and function of project | Need to know parameters and function of project |
| Bill of Materials | Not started | Finish complete B.O.M for project | Have not started on B.O.M yet | Need to know parameters and function of project | Need to know parameters and function of project |
| Impact and risk report | Not Started | Finish Impact and risk report for project | Not started on impact and risk report yet. | Need to know parameters and function of project | Need to know parameters and function of project |

Table 2: Project Status

## 1.5. References and File links

### 1.5.1. References (IEEE)

[References to for Block Diagrams]

### 1.5.2. File Links

[Datasheets]

## 1.6. Revision Table

| | |
|---|---|
| 11/12/2021 | Felipe: Updated Executive summary to reflect current design of the project |
| 11/12 | Ryan: Section 1 Writing and Structure based on peer review and instructor notes. Expanded Gap Analysis section. |
| 11/12/2021 | Felipe: Updated Timeline Description paragraphs |
| 10/29/2021 | Kevin: Added in table and bullet point list for team communications |
| 10/29/2021 | Felipe: New timeline |
| 10/29/2021 | Dennis: Reviewed document and worked on gap analysis |
| 10/29/2021 | Felipe: Include project partner contact information, move table from GAP analysis to timeline and add explanation in GAP analysis |
| 10/21/2021 | Felipe Orrico Scognamiglio: Proofreading and final touches (1.1 and 1.2), Section 1.4 (explanation of phases) |
| 10/21/2021 | Kevin Ho: Team Communications Protocols and Standards |
| 10/21/2021 | Dennis Kichatov: Executive summary (Intro) draft |
| 10/21/2021 | Ryan Christensen: Executive summary draft |
| 10/20/2021 | Kevin Ho: Initial draft of Timeline |
| 10/20/2021 | Dennis Kichatov: Initial draft Gap analysis |
| 10/17/2021 | Kevin Ho: Created document and outline |

Table 3: Revision Table Section 1

# 2. Requirements Impact and Risks

## 2.1. Requirements

Tektronix has provided a requirements writeup divided in five different priority groups, 1 being the most important and 5 the least important.

### 2.1.1. Project Requirement 1: The system will press a button

Engineering Requirement: The tool head will be able to interact and activate the buttons on the front of the oscilloscope, and properly trigger the buttons without touching the other buttons on the device.

Validation Process:

1. The command to push a certain button is sent by the user by selecting a saved button and pressing the designated button on the GUI to send the command.
2. The tool head will move to the correct button that is sent by the user.
3. The tool head will push the button.

### 2.1.2. Project Requirement 2: The system turns multiple sized knobs

Engineering Requirement: The tool head will be able to grip and turn a knob on the front of the oscilloscope, without touching the other knobs on the device.

Validation Process:

1. The command to turn a knob is sent by the user by selecting a saved knob and pressing the designated button on the GUI to send the turn command.
2. The tool head will move to the correct knob that is sent by the user.
3. The tool head will grip the corresponding knob.
4. The tool head will turn the knob.

### 2.1.3. Project Requirement 3: The system will have autonomous operation.

Engineering Requirement: The user will use the GUI to request movement to a label location or a button press and the system will move the toolhead accordingly.

Validation Process:

1. The user will send a command to move the device along the X, Y, or Z axis, using a saved label location by selecting a label location or button location and pressing the corresponding button in the GUI to send the command.
2. The user interface will interpret the commands and send the gcode.
3. The motor controller will follow the g-code received and move the mechanical frame to the designated positions.

### 2.1.4. Project Requirement 4: The system will allow a user to manually control the device.

The user will be able to manually control the system over a GUI and move the toolhead to its location, similar to jogging on a CNC Machine.

Validation Process:

1. The user will send a command to the system to move the device along the X, Y, or Z axis, by typing the gcode into the terminal on the GUI.
2. The user interface will relay the commands the the motor control block.
3. The motor controller will follow the gcode received and move the mechanical frame to the designated positions.

### 2.1.5. Project Requirement 5: The system will be able to toggle power to the stepper motors through the GUI.

Engineering Requirement: The user is able to toggle the relay powering the stepper motors of the mechanical structure by pressing the power button on the GUI.

Validation Process:

1. The user presses the power button located on the top left portion of the GUI.
2. The system will turn ON or OFF the relay powering the stepper motors.
3. The current state of the relay can be seen updated on the GUI

### 2.1.6. Project Requirement 6: The system will produce a video feed

Engineering Requirement: The system will collect a video feed from a device attached to the Raspberry Pi 4. The video feed will be accessible to the user in the user interface. The video feed can be recorded for future use.

Validation Process:

1. Inside the User Interface, the user will click on the toggle video feed button.
2. The user starts recording the video by typing in a file name on the text field and toggling the record video button.
3. Video starts being recorded to the desired filename.
4. The user stops the video recording by toggling the video recording button.
5. Videos are saved to the filename given by the user on the user interface folder.

### 2.1.7. Project Requirement 7: Fits in a defined size

Engineering Requirement: The mechanical system will fit inside a 22in height x 34in width x 20in depth space. This is to ensure that Tektronix will be able to fit the device on either a shelf or a desk with easy access.

Validation Process:

1. Using a measuring tape, the system width will be shown to be no greater than 34in.
2. Using a measuring tape, the system depth will be shown to be no greater than 20in.
3. Using a measuring tape, the system height will be shown to be no greater than 22in

### 2.1.8. Project Requirement 8: The system will display commands on screen during video playback

Engineering Requirement: The system will display the currently being executed command in the video feed as well as recordings.

Validation Process:

1. The user will press the toggle video button
2. The user will start a recording
3. The user will issue a command to the system
4. Upon completion of the command, the user will stop the recording
5. The user will check the recording for executed command

## 2.2. Design Impact Statement

### 2.2.1 Public Safety

#### 2.2.1.1 Impacts

This project addresses the need to increase productivity in Tektronix. The product cannot be a threat to anyone who is near or working with it. As technology improves, the products will be faster and more efficient. This can sometimes overshadow the thought of safety measures. Our project could be operating alongside an employee, it cannot pose a threat to them or any other machinery around it. The requirements also stated that the machine can be automated. With automation a bug can occur and the machine can behave differently from what was expected.

To reduce the damage of the bug an emergency stop button can be implemented or the device can be isolated during use. Thomas Arnold and Matthias Scheutz stated in *The “big red button” is too late: an alternative model for the ethical evaluation of AI systems,* that a button can give human control over the AI interface [1]. Even though our machine does not run full by itself, it could have a chance of malfunctioning during a command. A malfunction can cause damage to machines around it or people. A button will give the operator control over it, just in case any issues occur [1]. Another solution of this would be to have the project inside an enclosure, from Ken Meyers [2]. The main idea is to isolate the issue within a contained space where any damage is contained [2]. This will keep out any person to interact with the project while it is running. If a malfunction occurs it will contain the damage to the device inside the enclosure. Harming another machine would be bad but not as bad as hurting a human. The button to stop the machine would be the best solution because it gives the user a direct sense of control over the product. The button ideally would be a physical button that is on our project

that is in an easy spot for the user to reach and use. Another step forward is to design the button to have a virtual accessibility so the machine can run remotely and the user can stop it remotely.

#### 2.2.1.2 References IEEE

[1] Thomas Arnold, Matthias Scheutz, “The “big red button” is too late: an alternative model for the ethical evaluation of AI systems”, *springerlink.com.* [Online] Available: https://link.springer.com/article/10.1007/s10676-018-9447-7 [Accessed Oct. 27, 2021]

[2] Ken Meyers, “Faster, High Tech Machines Demand Enhanced Safety Precautions”, *proquest.com.* [Online] Available: https://www.proquest.com/openview/77c1a249ec43d3639c62d742aec23359/1?pq-origsite=gscholar&cbl=35812 [Accessed Oct 27, 2021]

#### 2.2.1.3 Revision Table

| | |
|---|---|
| 05/06/2022 | Kevin: Added section 2.2.1 |

## 2.2.2 Welfare Impacts

#### 2.2.2.1 Impacts

This automated solution can assist human testing of some of Tektronix’s products. During the COVID-19 pandemic, the need for ways to interact remotely greatly increased. According to BBC News, the robotization of work that was previously done by humans can help with social distancing and quality of life. “People will prefer to go to a place that has fewer workers and more machines” [1].

Automation can also come with some safety concerns. If the solution is not properly validated for safety, it can cause problems for human operators that could lead to injuries or worse. For that reason, the American National Standards Institute, the International Organization for Standardization, and others created safety requirements for all industrial robots and robot systems [2]. Since Tektronix plans to employ this solution as an alternative to testing the equipment in person, the implications of safety are greatly reduced. Of course, there are still minimum operating standards that must be met so that handling the equipment is safe and effortless.

As evidenced in section 2.2.4.1 of this report (Environmental Impacts), electronic waste is highly toxic to humans, plants, and livestock. Those toxins may cause problems with the brain, liver, and other organs. It could also lead to birth defects and congenital diseases. The particles released from the burning of electronic waste will also eventually settle on the soil and may contaminate causing a large range of undesirable outcomes such as altering pH levels of soil and water and temperature changes. Heavy metals, such as lithium, lead, and barium can

pollute the groundwater beneath the deposits and ultimately damage ecosystems. Recovery from this is highly unlikely. In order to mitigate the risks of this environmental impact, and consequently mitigate the health, safety, and welfare impacts, this project will avoid unnecessary use of potentially toxic materials and try to reduce the amount of materials (such as plastics on PCBs and 3D printed enclosures, toxic metals in PCBs, and materials that are environmentally expensive to produce such as the Aluminum used for the frame of the gantry) needed across all areas of the project.

#### 2.2.2.2 References IEEE

[1] Z. Thomas, “Coronavirus: Will covid-19 speed up the use of robots to replace human workers?,” *BBC News*, 18-Apr-2020. [Online]. Available: https://www.bbc.com/news/technology-52340651. [Accessed: 29-Oct-2021].

[2] “Robotics Standards,” *Robotics - Standards | Occupational Safety and Health Administration*. [Online]. Available: https://www.osha.gov/robotics/standards. [Accessed: 29-Oct-2021].

#### 2.2.2.3 Revision Table

| | |
|---|---|
| 05/06/2022 | Felipe: Added Sections 2.2.2.1, 2.2.2.2, and 2.2.2.3 |

### 2.2.3 Cultural and Social Impacts

#### 2.2.3.1 Impacts

The number one concern about automation is the effect it will have on jobs and what it means for low-skilled workers. But the impacts of automation are much broader than simply job stability. Automation has an impact within workplace culture, human machine relationships, and workplace effectiveness.

A report by the International Federation of Automatic Control, a collection of engineers and scientific societies put forth a report of the social and human-centered approach that automation has currently. [1] They argue that with the focus and strength being put into computer and technical-centric research, that we lose sight of the human-centric areas. “There is a too large polarisation between technically-oriented reasoning on one hand, and socially-oriented reasoning on the other in order to efficiently apply manufacturing technologies such as enterprise resource planning systems in less automated areas such as heath-care” [1]. Because of the technology currently in use, automated systems currently only serve the technical-centric workplace and research, often leaving behind the human aspect of machines. The methods and technologies developed for engineers, factories, and power plants, cannot be used within areas such as health-care, retail, or government, as the nature of work is so vastly different that trying to use the same autonomous system from one to another, doesn’t work. This

creates a problem that needs to be addressed, one where automated and machine systems are built to work with humans in what the paper calls “Human-Machines” [1].

But while the social issues, such as policy, labor, and workforce can be clearly defined and understood, the cultural aspect is a bit more invisibile, which can be a serious misstep. “There is sharpened global competition in business, where technology in itself is less seen as the prime weapon. It is rather the way it is being utilized, which depends on human resources: skills, creativity, values, commitment etc. that gives the (competitive) edge. Such factors have a cultural background and we thus have to start to look at culture as a competitive advantage.” [2] People are not just defined by the company they work for or the degree they have but through the culmination of their background and experiences. This is a key factor to working effectively, and is often why workplace culture is a major factor on an employees performance. Automated systems are no less affected by this, their implementation, and the way they integrate with a workplace needs to be carefully considered.

Being able to reach out and talk with the Tektronix engineers and get to know them has been a great way to understand the Tektronix work culture and the way they would interact with our tools. With the goal of building a system that supplements the Tektronix Engineers, creating a system that is accessible for a variety of engineers and offering thorough documentation, we hope that anyone who works with the system has the ability to learn our device.

#### 2.2.3.2 References IEEE

[3] Mayer, Frederique. ‘Social impact of automation trends and issues: an human centred systems engineering perspective” Sciencedirect.com. July 2008. Web. [Accessed Nov, 30.]
https://www.sciencedirect.com/science/article/pii/S1474667016397762

[4] Cernetic, Et. Al. “Revisiting the social impact of automation” IFAC 2002. Web. [Accessed Dec, 1.]
https://folk.ntnu.no/skoge/prost/proceedings/ifac2002/data/content/02899/2899.pdf

#### 2.2.3.3 Revision Table

| | |
|---|---|
| 05/06/2022 | Ryan Christensen: Added Sections 2.2.3.1 - 2.2.3.3 |

### 2.2.4 Environmental Impacts

#### 2.2.4.1 Impacts

Just like any other electric or electronic solution, this project is bound to have some environmental impacts. This project may or will impact the environment in the following ways (not limited to): Generation of electronic waste and energy consumption.
Energy consumption is a very important impact. With the increase of electronic devices across the globe, the impacts of energy generation are increasingly more evident. It is estimated that

by 2035, energy generation, will contribute to over 76% of global greenhouse gas emissions [1]. The environmental impacts of energy generation are not limited to just greenhouse gas emissions, there is also water usage that can be over one thousand one hundred gallons per kilowatt-hour for coal power plants or four thousand for geothermal power. And it does not stop there. When burning biofuels, there is a consumption of over nine hundred and seventy-three thousand liters per kilowatt-hour (Biofuels: soybean) of energy generated [2]. It is estimated that the costs of changing the way energy is produced are much higher than building more efficient products [1]. Knowing those factors, and also knowing that there is a high likelihood that this solution will stay effectively online for an average of twenty-four hours a day and seven days a week, will affect the design of this project so that it is the most efficient possible solution (based on the member's current knowledge and experience) for the sole purpose of avoiding wasteful use of energy.

The generation of electronic waste is a very important point to consider when designing a solution to a problem. Just like any other product, this solution has a limited lifecycle. "Unbeknownst to many consumers, electronics actually contain toxic substances - therefore they must be handled with care when no longer wanted or needed" [3]. It is bound by the physical constraints of the design and unavoidably will produce waste. In the next few paragraphs, I will explain the negative effects of electronic waste on the air, soil, water, and humans.

It is possible to see contamination in the air when the electronic waste is dismantled, shredded, or melted, this way releasing dust and or toxins, such as dioxins [3] (group of chemical compounds that are persistent organic pollutants in the environment), into the environment and not only pollute the air, but can cause health problems. One of the most decisive problems that the burn of electronic waste brings is the release of very small particles that can travel thousands of miles and pose health risks to humans or animals.

Soil contamination is also a very important topic when talking about the negative effects of electrical waste. The improper disposal of electronic waste to landfills can cause both heavy metals and flame retardants [3] to contaminate the soil and ultimately contaminate the underlying underground water, this way contaminating crops or animals that may use that water later on which can cause illness and even death. The particles released from the burning of electronic waste will also eventually settle on the soil and may contaminate causing a large range of undesirable outcomes such as altering pH levels of soil and water and temperature changes [3].

Just like evidenced before, the impacts of electronic waste in water are not small. Heavy metals, such as lithium, lead, and barium can pollute the groundwater beneath the deposits and ultimately damage ecosystems. Recovery from this is highly unlikely.

Lastly, as humans are inherently selfish, the negative impacts of electronic waste on humans. As evidenced in the paragraphs above, electronic waste is highly toxic to humans, plants, and livestock. Those toxins may cause problems with the brain, liver, and other organs. It could also lead to birth defects and congenital diseases.

For those reasons listed above and others, this project will aim to be the most efficient possible with the use of non-renewable resources such as heavy metals and plastics. This can be accomplished by reducing the size of PCBs and other components of the project that contain

those substances and materials, or even replacing them whenever possible for other less impactful or renewable materials.
With this policy, this team aims to try to prevent the contamination of the air, water, and soil by effectively reducing the sources of contamination that are present in the design and constructing the possible parts of the solution with renewable materials (such as biodegradable filament for all 3D printed portions of the project) and, if available, recycled ones (such as recycled aluminum for the frame, or refurbished/used components like Stepper Motor Drivers, or the Stepper Motors themselves).

#### 2.2.4.2 References IEEE

[1] J. Sandwood, “Climate change is pushing electrical engineers to focus on the environment,” *EEWeb*, 07-Jan-2019. [Online]. Available: https://www.eeweb.com/climate-change-is-pushing-electrical-engineers-to-focus-on-the-environment/. [Accessed: 29-Oct-2021].

[2] “Environmental impact of electricity generation,” *Wikipedia*, 28-Oct-2021. [Online]. Available: https://en.wikipedia.org/wiki/Environmental_impact_of_electricity_generation. [Accessed: 29-Oct-2021].

[3] “Waste & its negative effects on the environment,” *E*. [Online]. Available: https://elytus.com/blog/e-waste-and-its-negative-effects-on-the-environment.html. [Accessed: 29-Oct-2021].

#### 2.2.4.3 Revision Table

| | |
|---|---|
| 05/06/2022 | Felipe: Added section 2.2.4 |

## 2.2.5 Economic Impacts

#### 2.2.5.1 Impacts

The team’s project would be very beneficial for the economy of Tektronix and smaller companies. The reason for this is because our design will improve the speed of testing for smaller circuits and oscilloscopes. By using our system to automate the routine process of setting up the oscilloscope for testing we save a lot of time the engineers would spend not working on their circuit. Our system can even be used to stress test oscilloscopes and their buttons and knobs by having it autonomously push a button thousands of times until it breaks. This means that limit tests don't have to be tested by a person saving the company time and money. The problem with this lies in the article [6]. By implementing our system this way our design would essentially be replacing a person in their job.This would mean new personnel would have to be educated in our system, how it works and how to fix it. This will save the company money in the long run but initially production would slow down. The downside to this approach would be that some people would lose their current job or have to change jobs. This is

because our device removes the need for specialized people to test equipment. With the robot any person will be able to run a preloaded script for any oscilloscope without needing prior knowledge. This means companies can save money by laying off their tech tester occupations.

Another downside to using our system would be the break factor. According to tektronix[7] their new series 5 oscilloscopes cost $20,000-$30,000 and they want to use our system on these devices. Since we are taking the human factor away from the oscilloscope there is always a chance that our system can break the scope. The device could break the scope by pushing too hard on the buttons or by over rotating the knobs. Some knobs on an oscilloscope don't rotate 360 degrees and by mixing up the knobs the machine could break one. Another way an oscilloscope could be broken is by falling. From Tektronix, we know that they store some of their oscilloscopes in high places and when testing with the device, the machine could push too hard and thus knocking a scope over. This will be a costly replacement for Tektronix and would take a lot of time to get a new replacement.

#### 2.2.5.2 References IEEE

[6] Edgepoint Learning. What's The Real Cost of Training New Employees? Micheal Hanso. 10.29.2021.[Online]. Available: https://www.edgepointlearning.com/blog/cost-of-training-new-employees/.

[7] Tektronix. 5 Series MSO Mixed Signal Oscilloscope. 10.29.2021. [Online]. Available:https://www.tek.com/oscilloscope/5-series-mso-mixed-signal-oscilloscope.

#### 2.2.5.3 Revision table

| | |
|---|---|
| 05/06/2022 | Dennis: Added Economic Impact section 2.2.5 |

## 2.3. Risks

| Risk ID | Risk Description | Risk Category | Risk Probability | Risk Impact | Performance indicator | Responsible party | Action Plan |
|---|---|---|---|---|---|---|---|
| R1 | Vendor Delay | Time | Medium | H | Shipping times | Anyone who needs to order parts/components<br>- Ryan<br>- Dennis<br>- Kevin | To reduce the risk the team will plan ahead (order needed parts early.) |
| R2 | PCB schematic do not work | Technical | Low | H | Working PCB | Anyone who orders a PCB<br>- Ryan<br>- Dennis<br>- Kevin | To reduce the risk the team will verify PCB design and schematic with other group members before ordering. |
| R3 | Over budget | Cost | Low | M-L | Budget spreadsheet | Anyone who needs to purchase something for the project<br>- Ryan<br>- Dennis<br>- Kevin | To retain the problem the team will revise and change designs if needed to fit the budget. |
| R4 | Incompatible blocks | Technical | Low | H | Block combination | People whose block interfaces with another<br>- Ryan<br>- Kevin<br>- Dennis<br>- Felipe | To reduce the risk the group will decide during the design phase how blocks interface as a requirement for that block. Also communications/updates with other group members whose block is connected. |
| R5 | Project Partner Approval | Time | Low | M | Project Partner Updates | Group members who are assigned to check off blocks. | Get the schematic and design done early to send to the project partner this will reduce the likelihood for this |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | - Ryan<br>- Kevin<br>- Dennis<br>- Felipe | risk to happen. |
| R6 | Block Completion | Technical | Low | H | Group meetings | Every group member | To retain the problem the team will ask for help and advice if a problem occurs that the one member can't solve. |

Table 4: Risk Analysis

## 2.4. References and File Links

### 2.4.1. References

### 2.4.2. File Links

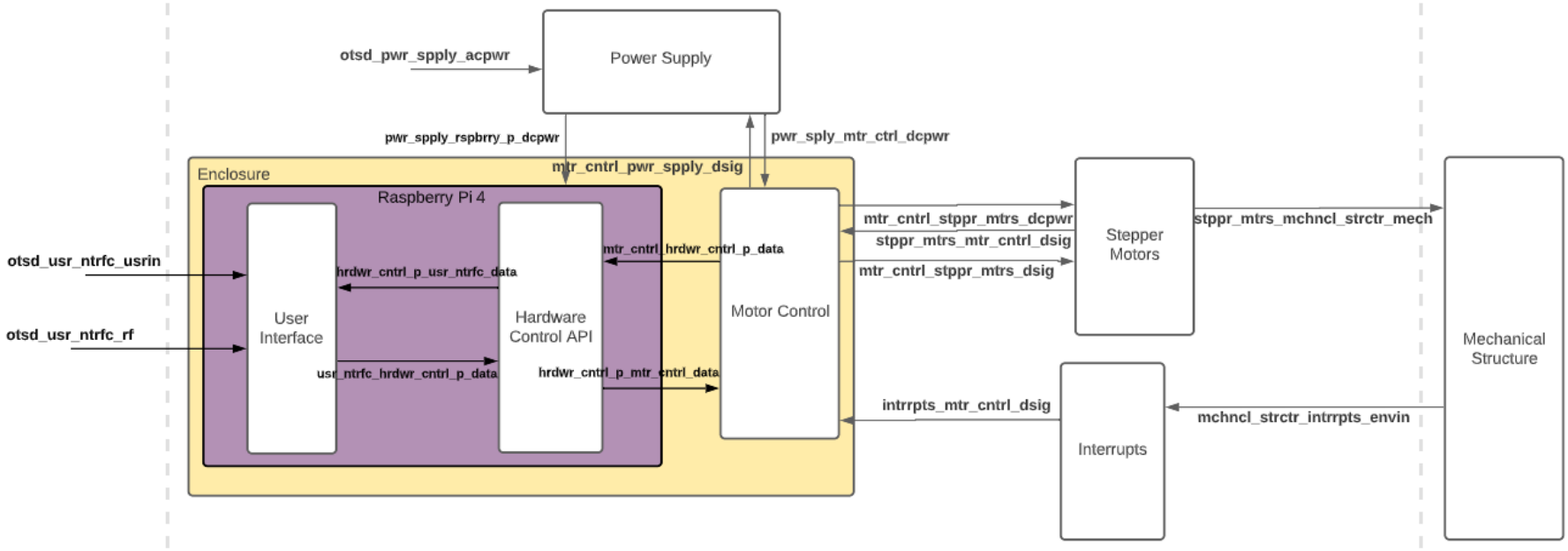
## 2.5. Revision Table

| | |
|---|---|
| 05/06/2022 | Kevin: Added section 2.2.1 |
| 05/06/2022 | Dennis: Updated Section 2.2.5 |
| 05/06/2022 | Felipe: Updated Sections 2.2.4 and 2.2.2 |
| 05/03/2022 | Ryan: Updated Project Requirements Section 2.1 to match Section 5 |
| 05/03/2022 | Felipe: Updated Project Requirements Section 2.1 to match Section 5 |
| 12/03/2021 | Felipe: Update Project Requirement 2.1.6 |
| 12/03/2021 | Ryan: Update priorities to fit the format needed & peer reviews |
| 12/03/2021 | Dennis: Update priorities to fit the format needed |
| 12/03/2021 | Kevin: Update priorities to fit the format needed |
| 11/12/2021 | Kevin: Update Priority 2, Risk action plan update |
| 11/12/2021 | Dennis: Peer review edits |
| 11/12/2021 | Ryan: Peer Review Edits to System Requirements |
| 11/12/2021 | Felipe: Update Priority 1 Requirements |
| 11/12/2021 | Felipe: Replaced percentages to Low/Medium/High in section 2.3 Risks |
| 10/29/2021 | Felipe: Included requirements checklist |
| 10/29/2021 | Kevin: Risk analysis draft, verification for checklist |

Table 5: Revision Table Section 2

# 3. Top-Level Architecture

## 3.1. Block Diagram



## 3.2. Block Descriptions

### 3.2.1 User Interface:

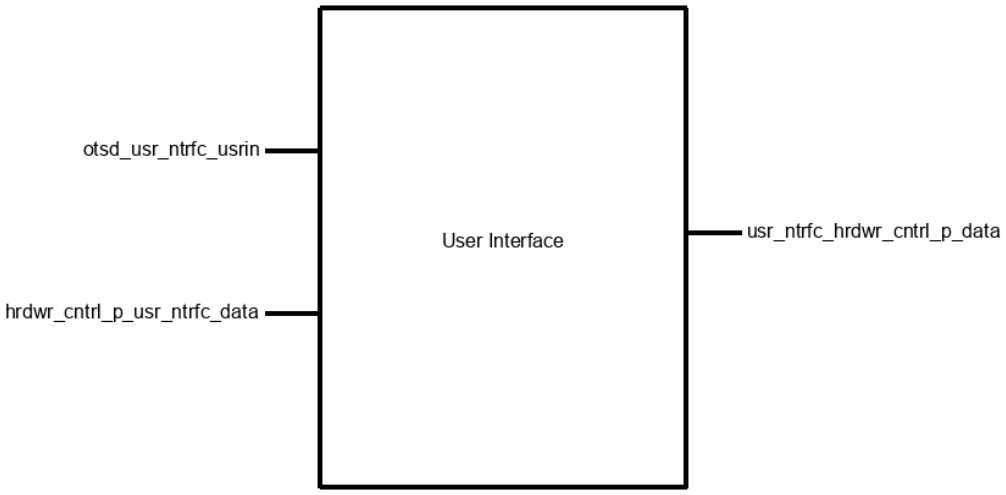



Figure 3: User Interface Block

The user interface receives user input through direct interaction with the user interface The user interface interacts with the Hardware Control API by translating the user inputs into usable data that is then sent to Hardware Control API for translation and serial forwarding. The user input can be considered as raw or sanitized input based on the source. The user is able to use preset commands by pressing a button in the User Interface, typing commands in the terminal, loading a script file, and loading a labels file. The user interface receives information from the user or the Hardware Control API (henceforth regarded as HCAPI). The input from the HCAPI consists of positional data, values returned from the microcontroller and other information that is available. The user input consists of keyboard input through a command-line interface available within the User interface, mouse clicks in buttons available within the User interface, Files containing positional data for buttons and knobs as well as knob height and diameter, and GCode scripts.

### 3.2.2 Power Supply:

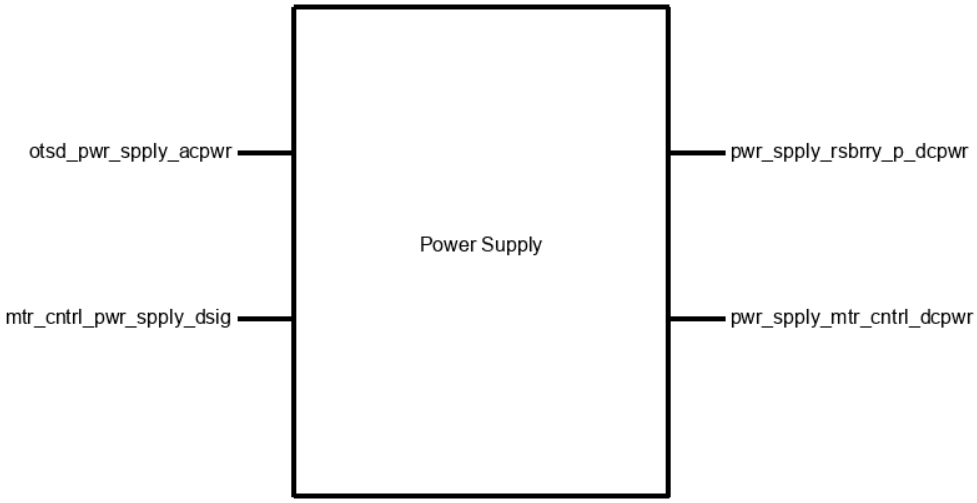



Figure 4: Power Supply Block

Using 120 AC Voltage power supplied from a wall outlet. The team will use a switching power supply from Stepperonline to regulate the AC voltage into usable 12V DC voltage. The switching power supply takes in 120VAC from the wall outlet and outputs 12VDC. From there the team will use custom DC to DC converters to regulate the voltage down and to be able to adjust the output current and power. The converters will need to lower the 12V into a workable 3.3-5 Volts, 3.5 Amps and 18 Watts to power the Raspberry Pi and microcontroller. Coming from the power supply and straight to the stepper motors is 12-16 Volts to power the stepper motors. A relay will act like a gate on the microcontroller and when there is no signal the power supply will be cut off from the microcontroller. The relay will form a digital kill switch to turn off the system. The block champion for the power supply is Dennis Kichatov.

### 3.2.3 Hardware Control API:

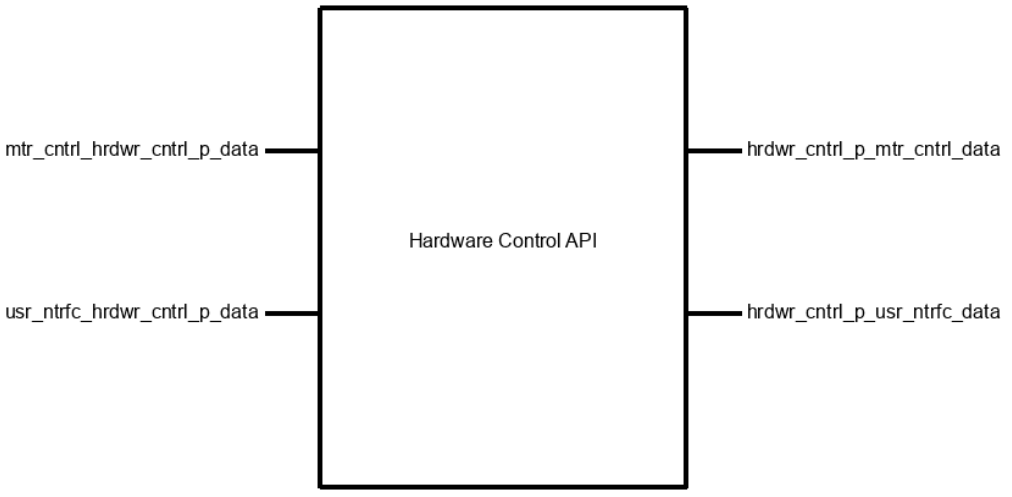



Figure 4: Motor Control API Black Box Diagram

The Hardware Control API receives the encoded user interface commands. The user data is then translated into GCode and put into a transmission queue. The Hardware Control API then goes through the transmission queue and sends the data packet to the motor control block. Upon receiving and completing the task, the motor control block will send a confirmation code back to the Hardware Control API announcing that it is ready to receive another data packet. The Hardware Control API allows the abstraction of the connection between the User Interface (or another program that uses this API) and the Motor Control block that runs a version of GRBL HAL. Usually, when interfacing with GRBL the user would need to directly send GCode or other configuration commands to the microcontroller, instead, with this API, the user is able to easily set up and move the payload of the gantry system with a simple custom implementation in python.

### 3.2.4 Motor Control:

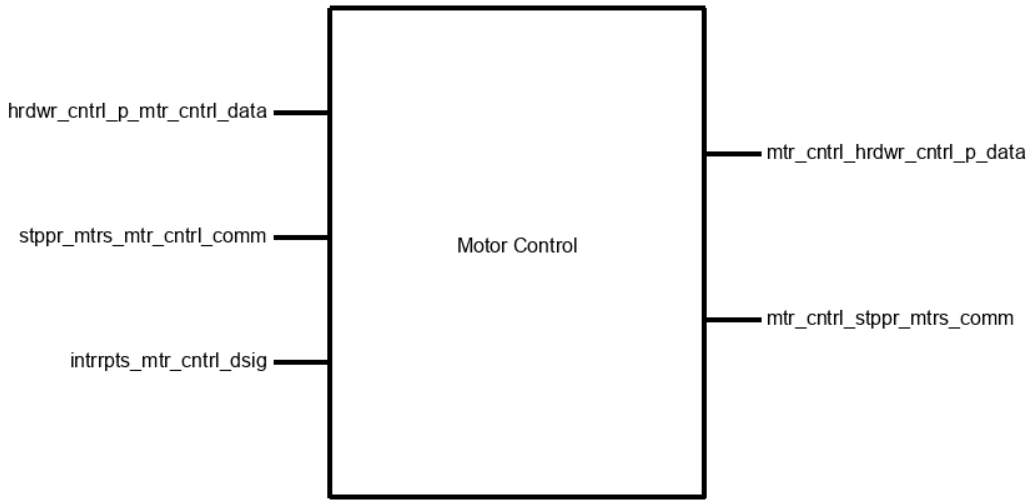



Figure 5: Motor Control Block

The motor control block receives data from the Raspberry Pi. The data received will be in gcode. It will use the data with GRBL HAL to send data to the motor drivers within the block to move the stepper motors to a xyz coordinate. The Motor controller will also route the voltage needed for each motor driver. The motor driver will also receive updates of the status of the motors through a digital signal. This signal is then passed on to the raspberry pi to update the user interface. This will be used to keep track of the state of the motors. The interrupts will send a digital signal to the motor controller. This signal will be from limit switches so the motor controller will know the limits of the mechanical structure. The block champion for Motor Control is Kevin Ho.

### 3.2.5 Interrupts

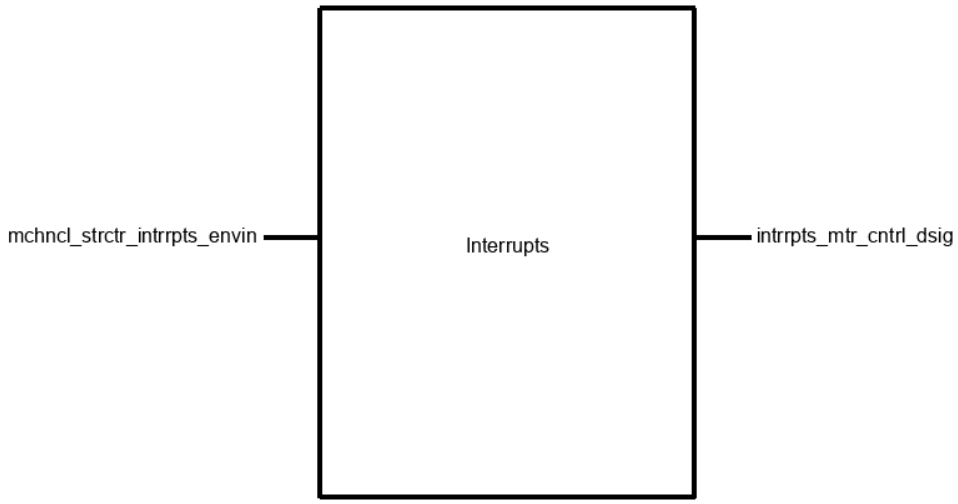



Figure 6: Interrupts Block

The interrupts block will be limit switches that will send a digital signal to the motor controller. The limit switches will act as boundaries for the motors and system. When the mechanical structure hits the limit switch, the switch will send a 1 to the motor controller to signal the bounds of the structure. The block champion for Interrupts is Kevin Ho.

### 3.2.6 Mechanical Structure:

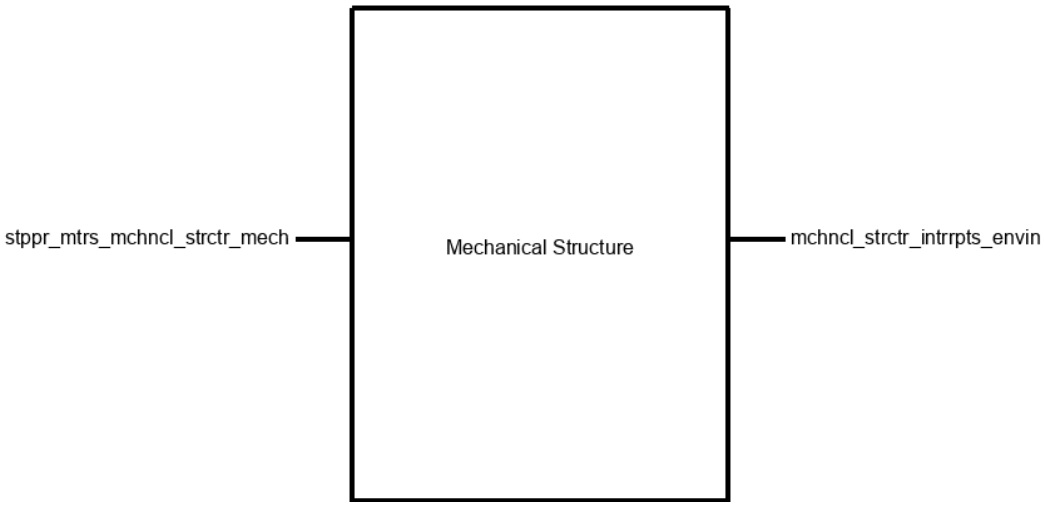



Figure 7: Mechanical Structure Block

The mechanical structure is the physical device that interacts with the oscilloscope. The mechanical structure will take the rotational movement of the stepper drivers on each axis and turn it into linear motion on linear rails. Through a series of belts and rails, the turning of each stepper motor will move the tool head in 3D space allowing the robot to move in each direction. The movement on each axis, including the toolhead, will be monitored by limit switches which will be used to notify the motor controller if the tool has been moved too far in either of the 4 directions. The champion for the Mechanical Structure is Ryan Christensen.

### 3.2.7 Enclosure:

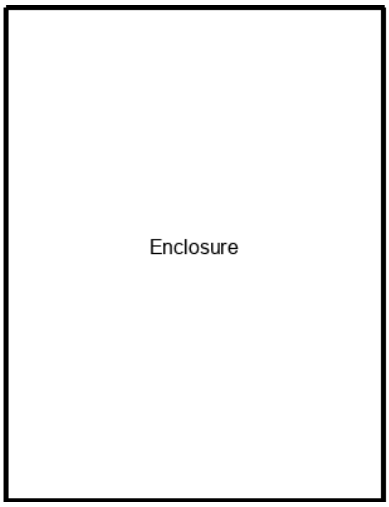



Figure 8: Enclosure Block

The enclosure for the system will keep all the circuitry and wiring together This enclosure will make sure that if the structure has to be moved all the wiring and PCB boards won't break or move. This enclosure will make the wiring easy to follow and will make the project look more presentable. The block champion for the Enclosure is Dennis Kichatov.

### 3.2.8 Stepper Motors:

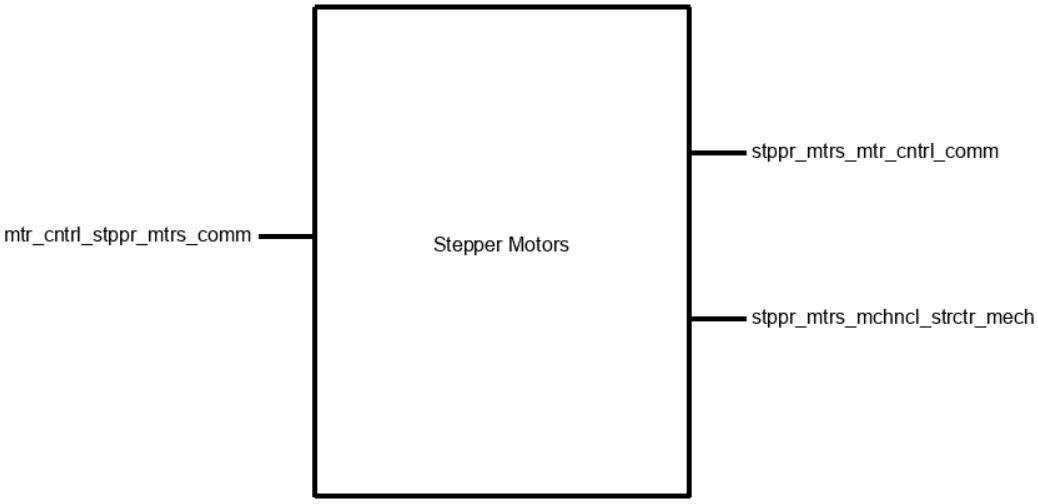



Figure 9: Stepper Motors Block

The Stepper Motors block includes both the Stepper Motor Drivers on the Motor Control PCB and their corresponding motors on the mechanical device. Taking in 12V DC Power, the drivers will convert the digital signal received from the Motor Controller board and convert it into pulses and current inside the motors. These currents and pulses operate the stepper motors on the gantry, which will in turn move in the corresponding axis in the direction and amount as directed by the driver. The champion for the Stepper Motors is Ryan Christensen.

## 3.3. Interface Definitions

### 3.3.1 User Interface

| 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? |
|---|---|---|
| **otsd_usr_ntrfc_usrin : Input** | | |
| Other: User click on button or write GCode in User Interface. | The Mouse and Keyboard are connected to the Raspberry Pi. Naturally, any GUI that is running on the Raspberry Pi is able to receive inputs from the keyboard and mouse. | The Raspberry Pi is able to read the information sent from the mouse and keyboard. The GUI should be able to receive this data whenever is selected as a window. The GUI has multiple buttons. Each button is clickable. The GUI also has a text box that will serve as a terminal that will accept written commands. |
| Other: Script File | User is able to load a script file containing a list of GCode commands to be sent to the microcontroller. | The GUI is able to load and run the script. Information about execution is added to the terminal. |
| Other: Labels File | User is able to load a file containing a list of Labels (.csv format) to be used as macros. | The GUI is able to load and interpret the list of labels and move the payload when requested. |
| **hrdwr_cntrl_p_usr_ntrfc_data : Input** | | |
| Messages: Hardware Confirmation and Location Data | After receiving a request, the API will return send back the requested data to the user. | The API is able to receive the string from the microcontroller, parse the necessary values from it, and return it to the user. |

| | | |
|---|---|---|
| Other: List of character arrays for x, y, z position | The work position, and machine positions are reported as x, y, and z positions. | The positional data is parsed by the GST and returned to the API as a list of strings, each containing the x, y, and z coordinates. |
| Other: Character array containing machine status | The machine status is reported as “Idle”, “Run”, among other values. All those values are strings sent by the microcontroller. | The machine status data is parsed by the GST and returned to the API as a single string to the User. |
| **usr_ntrfc_hrdwr_cntrl_p_data : Output** | | |
| Messages: Translated input from the user interface to Hardware Control API (button pushes) | After the user interfaces with the GUI, sanitized data is sent to the API to be interpreted and transmitted to the Motor Control block. The API is capable of translating raw positional data such as x, y, and z and feed rate F to Gcode commands that are queued in the GST. | Received data is understood and an appropriate response is triggered. If, for example, the data informs that the payload should move, relative to the current position, 3 cm on the X-axis, the API translates it to GCode and adds it to the broadcast queue. |
| Other: Sends request to set text over video stream, and take frame captures | The API is capable of setting text over the video stream or taking snapshots. Text is automatic, but frame captures require the user to name the file and press a button. | The API is able to request the VST to capture the current frame as a jpeg image or add text over the streaming video. (this should be a simple function call) |
| Other: Raw GCode commands to be sent to microcontroller (input from the terminal, script, or labels) | The API is capable of translating raw positional data such as x, y, and z and feed rate F to Gcode commands (labels).<br>Terminal and Script commands are sent without checking (unsanitized). | The API is able to generate commands to grblHAL that accept incremental or absolute coordinates and feed rate (from labels).<br>The API is capable of receiving Gcode commands from the terminal or script to send to the microcontroller. |

### 3.3.2 Hardware Control API

**hrdwr_cntrl_p_mtr_cntrl_data : Output**

| | | |
|---|---|---|
| Other: Sends one GCode command at a time | The API is expected to send only GCode commands and other GRBL HAL specific commands to the microcontroller. As per the design, the API will send only one GCode command at a time while waiting for a response from grblHAL. | The microcontroller receives the transmission and sends back a confirmation. If data was requested, data is sent back as well. |
| Other: Data Character Arrays | The API will send strings to the microcontroller that are essentially character arrays through serial. | Both grblHAL and the API expect the use of strings to communicate with each other. The API can generate and send, through serial, messages to the microcontroller. |
| Protocol: USB | The microcontroller is connected to the raspberry pi through USB (serial) | The raspberry pi has multiple USB ports and is able to connect to the microcontroller without a problem. With the use of the pyserial library, the API is able to interact with the USB connection and send and receive commands to the microcontroller. |

**hrdwr_cntrl_p_usr_ntrfc_data : Output**

| | | |
|---|---|---|
| Messages: Hardware Confirmation and Location Data | After receiving a request, the API will return send back the requested data to the user. At this point, the data is limited to Status confirmations and Locational data. | The API is able to receive the string from the microcontroller, parse the necessary values from it, and return it to the user. |
| Other: List of character arrays for x, y, z position | The work position and machine position are reported as x, y, and z positions. | The positional data is parsed by the GST and returned to the API as a list of strings, each containing the x, y, and z coordinates. |

| | | |
|---|---|---|
| Other: Character array containing machine status | The machine status is reported as “Idle”, “Busy”, among other values. All those values are strings sent by the microcontroller. | The machine status data is parsed by the GST and returned to the API as a single string. |

**mtr_cntrl_hrdwr_cntrl_p_data : Input**

| | | |
|---|---|---|
| Other: Sends one GCode command at a time | The motor controller is expected to send only confirmations and positional data to the Raspberry Pi. | The API receives the transmission and is able to process the data. If, for example, positional data is expected to be received, it will properly process and return to the user. |
| Other: Data Character Arrays From and to the µController to the Raspberry Pi 4 | The microcontroller will send positional data and confirmations to the API. That information is sent as a string that is essentially a character array through serial. | Both grblHAL and the API expect the use of strings to communicate with each other. The API can generate and send serial messages to the microcontroller. |
| Protocol: USB | The microcontroller is connected to the raspberry pi through USB (serial) | The raspberry pi has multiple USB ports and is able to connect to the microcontroller without a problem. With the use of the pyserial library, the API is able to interact with the USB connection and send and receive commands to the microcontroller. |

**usr_ntrfc_hrdwr_cntrl_p_data : Input**

| | | |
|---|---|---|
| Messages: Translated input from user interface to Hardware Control API | After the user interfaces with the GUI, sanitized data is sent to the API to be interpreted and transmitted to the Motor Control block. | Received data is understood and an appropriate response is triggered. If, for example, the data informs that the payload should move, relative to the current position, 3 cm on the X-axis, the API translates it to GCode and adds it to the broadcast queue. |

| | | |
|---|---|---|
| Other: Receives usable user input to request video frames, take snapshots, or set text over video feed | The API is capable of sending back to the user raw video frames (usually as a NumPy array), set text over the video stream, or taking snapshots. | The API is able to request the VST to send the latest frame as a pixel array, capture that frame as a jpeg image or add text over the streaming video. |
| Other: Receives usable user input information to translate to GCode commands and add to daemon queue | The API is capable of translating raw positional data such as x, y, and z and feed rate F to Gcode commands that are queued in the GST. | The API is able to generate Jogging commands to grblHAL that accept incremental or absolute coordinates and feed rate. The command is then added to the send queue. |

### 3.3.3 Mechanical Structure

**MCHNCL_STRCTR_INTRRPTS_ENVIN (X, Y, Z) : Output**

| | | |
|---|---|---|
| Description: Axis limit switch, used to home device and alarm status for Motor Controller | The limit switches will be used to check if the system is at the boundaries where we can use that information to alarm the system and home it. | The limit switches will be secured at the boundaries of the system. When the switch is closed the signal will go to the microcontroller to either stop the machine or home it. |
| Logic Level:0.3-0.0V, 5-4.7V | The limit switch will produce a signal logic high or low, 5V or 0V, depending if the structure hits the limit switch. | The common port on the limit switch will be connected to ground. The internal pull up resistor will be on to produce a 5-4.7B signal when the switch is hit. The switches are daisy chained in a way where ground will be going through the switches unless a switch is hit. |
| Data: CLOSED in normal state, OPEN when limit switch is active | The limit switch should be active when there is no physical contact on the switch. When the switch is pushed down the normal state should occur to produce the 5V signal. | The limit switch can be hooked up to be normally closed. This will create the needed interaction. |

**STPPR_MTRS_MCHNL_STRCTR_MECH: Input**

| | | |
|---|---|---|
| Minimum Linear | The system needs to | The default setting for GRBLHal was |

| | | |
|---|---|---|
| Speed: 1in/Sec | respond to an input in a timely manner, while retaining precision. Using a controlled speed like this helps to avoid unnecessary Jerk or Acceleration which may cause missed steps. | about 6in/sec. By staying at the default we give ourselves headroom of 4-5in/second in case reliability or precision becomes an issue. |
| Minimum Precision +/- .5in | The system will be working with | Using a direct belt system we can directly control the distance traveled by the motor through the number of rotations. A single step is equal to about .15mm of travel in the linear direction. Well below the .5in minimum. |
| Minimum Force Output; 20oz | The motor needs to be strong enough to move the device. | Previous testing has shown that the gantry required 6-10ozs to pull the gantry back and forth along the X axis. |

### 3.3.4 Stepper Motors

**MTR_CNTRL_STPPR_MTRS_COMM: Input**

| | | |
|---|---|---|
| Protocol: Step-Direction | The stepper driver chosen uses step and direction to interface with it [2]. | GRBL HAL[1] sends the step and direction signals given the G code where the output pins are connected to the stepper driver. |
| Logic Level: 3.3V, 0V | The stepper driver uses 3.3V and 0V for the direction [2]. The output pin from the µController does not provide 3.3V exactly so it should fall around it. | The stepper driver will need 3v3 for going one direction and 0 going the other. The µController outputs 3v3 for logic high from the datasheet. |
| Messages: Direction to spin motor - Step or Hold motor direction | There will be 4 stepper motors and stepper drivers and each will be routed to a specific pin on the µController. One motor for each | Step and direction pins for each stepper are mapped for different axes in grbl HAL[1]. The µController produces a voltage on the step pin depending on the speed. If there is 0 voltage the motors will stop. |

| | | |
|---|---|---|
| | axis and one for the toolhead. The step will change, increase or stay zero to tell the stepper to move forward or stop. | |

**STPPR_MTRS_MCHNL_STRCTR_MECH: Output**

| | | |
|---|---|---|
| Minimum Linear Speed: 1in/Sec | The system needs to respond to an input in a timely manner, while retaining precision. Using a controlled speed like this helps to avoid unnecessary Jerk or Acceleration which may cause missed steps. | The default setting for GRBLHal was about 6in/sec. By staying at the default we give ourselves headroom of 4-5in/second in case reliability or precision becomes an issue. |
| Minimum Precision +/- .5in | The system will be working with | Using a direct belt system we can directly control the distance traveled by the motor through the number of rotations. A single step is equal to about .15mm of travel in the linear direction. Well below the .5in minimum. |
| Minimum Force Output; 20oz | The motor needs to be strong enough to move the device. | Previous testing has shown that the gantry required 6-10ozs to pull the gantry back and forth along the X axis. |

## 3.3.5 Interrupts

**INTRRPS_MTR_CTRL_DSIG: Output**

| | | |
|---|---|---|
| Protocol: Digital Input | Each pair of limit switches will be hooked up to a digital pin on the microcontroller | The switches will produce a logic level high or low which a digital pin can take in. |
| Logic Level: 5-4.7V, 0.3-0.0V | The signal that is needed for the microcontroller is 5V for logic 1. | The common port on the limit switch will be connected to ground. The internal pull up resistor will be on to produce a 5-4.7B signal when the switch is hit. The switches are daisy chained in a way where ground will be going through the switches unless a switch is hit. |

| | | |
|---|---|---|
| Data: Limit Switch, depending on which axis switch is activated. | Since there will be more than one limit switch the data of each limit switch will be sent to the microcontroller. | The motor controller will be able to differentiate which limit switch is hit though digital input by knowing which way the tool head is moving. |

**MCHNCL_STRCTR_INTRRPTS_ENVIN (X, Y, Z) : Input**

| | | |
|---|---|---|
| Description: Axis limit switch, used to home device and alarm status for Motor Controller | The limit switches will be used to check if the system is at the boundaries where we can use that information to alarm the system and home it. | The limit switches will be secured at the boundaries of the system. When the switch is closed the signal will go to the microcontroller to either stop the machine or home it. |
| Logic Level:0.3-0.0V, 5-4.7V | The limit switch will produce a signal logic high or low, 5V or 0V, depending if the structure hits the limit switch. | The common port on the limit switch will be connected to ground. The internal pull up resistor will be on to produce a 5-4.7B signal when the switch is hit. The switches are daisy chained in a way where ground will be going through the switches unless a switch is hit. |
| Data: CLOSED in normal state, OPEN when limit switch is active | The limit switch should be active when there is no physical contact on the switch. When the switch is pushed down the normal state should occur to produce the 5V signal. | The limit switch can be hooked up to be normally closed. This will create the needed interaction. |

### 3.3.6 Motor Control

**MTR_CNTRL_HRDWR_CNTRL_P_DATA : Output**

| | | |
|---|---|---|
| Protocol:USB | The connection between the raspberry pi and µController is serial. | The raspberry pi can communicate through usb using the terminal and the black pill or the STM32F4 can as well from the datasheet section 3.27. |
| Other:Data Character Arrays From and to the | Grbl [1] will accept a string or a character array and decode that | When using the serial monitor the µController can receive arrays and send arrays through serial. |

| | | |
|---|---|---|
| µController to the Raspberry Pi 4 | for g code and other commands. | |
| Other:Send one array of character containing current states | Grbl can receive a command that reports the current state of the process and gives feedback that commands are received [1]. | Grbl HAL sends updates through serial or usb [1]. These messages are put into one array. Grbl HAL will send feedback messages to acknowledge a command is sent and will send its status if asked for. |

**HRDWR_CNTRL_P_MTR_CNTRL_DATA: Input**

| | | |
|---|---|---|
| Protocol:USB | The connection between the raspberry pi and µController is serial. | The raspberry pi can communicate through usb using the terminal and the black pill or the STM32F4 can as well from the datasheet section 3.27. |
| Other: Data Character Arrays | Grbl [1] will accept a string or a character array and decode that for g code and other commands. | When using the serial monitor the µController can receive arrays and send arrays |
| Other:Sends one GCode command at a time | Grbl [1] can decode one g code per command. | GRBL HAL[1] can receive one GCode and send the right signal to the drivers. |

**MTR_CNTRL_STPPR_MTRS_COMM: Output**

| | | |
|---|---|---|
| Protocol: Step-Direction | The stepper driver chosen uses step and direction to interface with it [2]. | GRBL HAL[1] sends the step and direction signals given the G code where the output pins are connected to the stepper driver. |
| Logic Level: 3.3V, 0V | The stepper driver uses 3.3V and 0V for the direction [2]. The output pin from the µController does not | The stepper driver will need 3v3 for going one direction and 0 going the other. The µController outputs 3v3 for logic high from the datasheet. |

| | | |
|---|---|---|
| | provide 3.3V exactly so it should fall around it. | |
| Messages: Direction to spin motor - Step or Hold motor direction | There will be 4 stepper motors and stepper drivers and each will be routed to a specific pin on the µController. One motor for each axis and one for the toolhead. The step will change, increase or stay zero to tell the stepper to move forward or stop. | Step and direction pins for each stepper are mapped for different axes in grbl HAL[1]. The µController produces a voltage on the step pin depending on the speed. If there is 0 voltage the motors will stop. |

**INTRRPS_MTR_CTRL_DSIG: Input**

| | | |
|---|---|---|
| Protocol: Digital Input | Each pair of limit switches will be hooked up to a digital pin on the microcontroller | The switches will produce a logic level high or low which a digital pin can take in. |
| Logic Level: 5-4.7V, 0.3-0.0V | The signal that is needed for the microcontroller is 5V for logic 1. | The common port on the limit switch will be connected to ground. The internal pull up resistor will be on to produce a 5-4.7B signal when the switch is hit. The switches are daisy chained in a way where ground will be going through the switches unless a switch is hit. |
| Data: Limit Switch, depending on which axis switch is activated. | Since there will be more than one limit switch the data of each limit switch will be sent to the microcontroller. | The motor controller will be able to differentiate which limit switch is hit though digital input by knowing which way the tool head is moving. |

### 3.3.7 Enclosure

**otsd_enclsr_other**

| | | |
|---|---|---|
| Dimensions | The enclosure will be large enough to fit all electronics and leave room for wiring. | Took dimensions of Raspberry Pi 4, Black Pill and switching power supply. |
| Inside | The enclosure will encompass the Raspberry Pi 4, Black Pill, step down voltage converter and stepper PCB. In a separate enclosure the switching power supply will be stored. | There will be stand offs for each component. |
| Enclosed and attached to the frame. | Both enclosures will be sealed from the top and PCBs bolted to the enclosure. | The lid will fit the enclosure and nuts and bolts will be used to attach the enclosure to the frame. |

### 3.3.8 Power Supply

**Otsd_pwr_spply_acpwr**

| | | |
|---|---|---|
| Inominal: 8.0A | This value is based on the description of the switching power supply. | From the description of the switching power supply. |
| Ipeak: 8.5A | This value is based on the description of the switching power supply. | From the description of the switching power supply. |
| Max Power:100W | This value is based on the description of the switching power supply. | From the description of the switching power supply. |
| Vnominal: 12V DC | The team needs 12 DC Volt output for the motor drivers. | From the description of the switching power supply. |
| Voltage Input: 84-264 VAC | This value is based on the description of the switching power supply. The power supply will use 120VAC from the wall socket. | From the description of the switching power supply. |

**pwr_spply_rspbrry_p_dcpwr**

| | | |
|---|---|---|
| Inominal: 3.0A | This is the current needed to help power the Raspberry Pi. | This is the current needed to power the Raspberry Pi. The inductor will help output 3 Amps |
| Ipeak: 3.5A | This is the peak of the current that is outputted from the 5 Volt converter. The Raspberry Pi 4 needs about 18 Watts of power. | The USB-C is able to handle around 3.0 Amps. So 3.2A will be the max current input. |
| Output Connection: USB-C | The converter will connect to the Raspberry Pi using one of these connections. | This is the required connection for the Raspberry Pi. |
| Vmax: 5.5V | This will be the maximum voltage that will be input to the Raspberry Pi. | The Raspberry Pi is able to withstand 5.5 Volts in case the voltage fluxuates. 5.5 Volt will be the max. |
| Vmin: 5V | This will be the minimum voltage requirement. Going lower than 5 Volts will dampen results. | The Raspberry Pi takes in 5 Volts of current. |

**mtr_cntrl_pwr_spply_dsig**

| | | |
|---|---|---|
| Imax: 100mA | This is the maximum current that flows through the relay. | This is the maximum current that flows through the relay. |
| Microcontroller connection | This is a wire that connects to the input of the microcontroller | The datasheet for the relay explains how to connect it. |
| Motor driver connection | This is a wire that connects to the input of the motor driver | The datasheet for the relay explains how to connect it. |
| Signal Pin: pulled high | The signal pin will connect to a port on the Raspberry Pi 4. When the signal is pulled high the microcontroller or motor drivers will disconnect from the power supply. | The Raspberry Pi 4 port will send a signal to switch and turn the microcontroller or motor divers “on” or “off”. |

## 3.4. References and File Links

### 3.4.1. References (IEEE)

[1] “grblHAL/STM32F4xx”*Github.* 18 Feb, 2022. [Online] https://github.com/grblHAL/STM32F4xx [Date Accessed 18, Feb 2022]

[2] “DRV8825 Stepper Motor Driver Carrier, High Current” *Pololu.* 18 Feb, 2022. [Online] https://www.pololu.com/product/2133 [Date Accessed 18, Feb 2022]

### 3.4.2. File Links

[3] STM32 Datasheet
https://www.st.com/resource/en/datasheet/stm32f411ce.pdf

## 3.5. Revision Table

| | |
|---|---|
| 04/22/22 | Updated Block Diagram |
| 03/06/22 | Ryan Updated Sections 3.3.3 and 3.3.4 |
| 03/06/22 | Felipe: Removed old information, updated sections 3.3.1 and 3.3.2, and created headers for other blocks |
| 12/03/21 | Felipe: Updated and reviewed block interface |
| 12/03/21 | Kevin: Updated name scheme and reviewed block interface |
| 12/03/21 | Dennis: Updated and reviewed block interface |
| 12/03/21 | Ryan: Reviewed peer review edits and cleaned up definitions |
| 11/19/21 | Felipe Orrico: Interface Definitions and Block Descriptions |
| 11/19/21 | Ryan Christensen: Interface Definitions formatting and filling in, block diagram revision. Mechanical and Stepper block definitions. |
| 11/19/21 | Kevin: Interface definition and block descriptions |
| 11/19/21 | Dennis: Worked on Block Definitions and Block Interface |

Table 3: Revision table for section 3

# 4. Block Validations

## 4.1. Mechanical Frame - Ryan Christensen

### 4.1.1. Block Overview

The mechanical frame is the physical structure of the device. The purpose of this block is to translate the rotational of the stepper motors, in the Stepper Drive block, into movement to manipulate the oscilloscopes under test. To achieve this, we have created the following design, inspired by a Core-XY 3d-Printer, which uses two motors in conjunction to move the toolhead in the XY Direction. This “Core-ZY” has the advantage of lowering the center of gravity along the gantry. Improved precision over cartesian style devices which use three motors independent of each other. Reduced weight by moving the stepper from the Z rail, onto the gantry frame. Although this introduces more complexity within the design, we believe that the benefits of using a Core-ZY will yield improved results and higher precision necessitated for this project.

### 4.1.2. Block Design

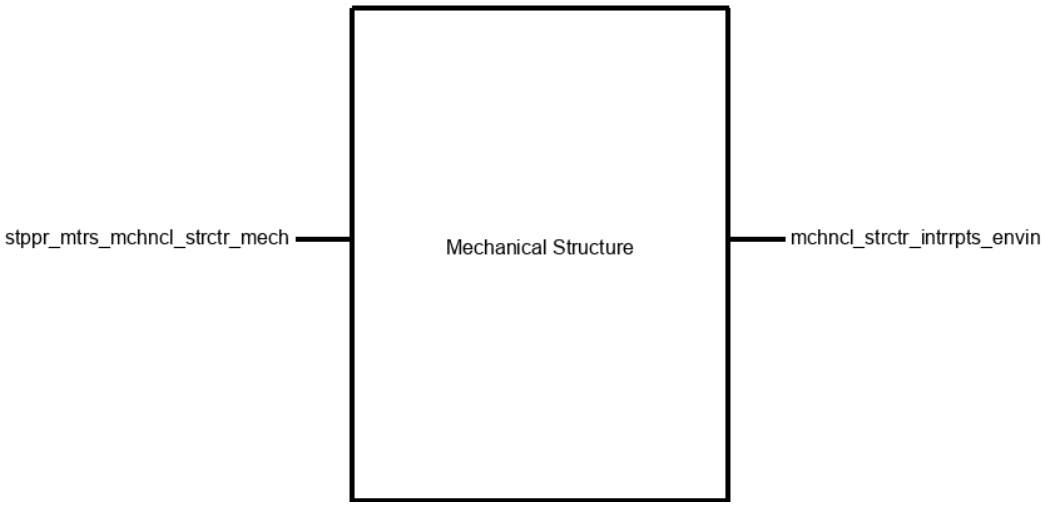



Figure 1. Black Box and Block Interface

### 4.1.3. Block General Validation

The most significant design consideration was how we would be able to interact with an oscilloscope’s button and knobs, with a high level of precision to make sure that when the device is active we get accurate measurement and tool placement. The second design consideration given to us by Tektrnoix is the capability to work around oscilloscopes of differing sizes as given to us by Tektronix. The largest Oscope size we will need was given to us, 7.5in depth x 12in height x 17.5in width. Our final design consideration for this block was how much space was available for the device itself. While we needed to fit different sized devices, it needed to be done while still retaining a compact form factor to fit on Tektronix shelves or desks.

Tektrnoix offered us a shelf size that would fit a device no larger than, in inches, 20in depth x 22in height x 34in width.

To fulfill these requirements the team chose between two approaches. The first was a robotic arm and the second was a gantry system. We decided to use a gantry as it would require the least amount of space on a desk when working with an oscilloscope, was much easier to design and order and could be built much faster than what was required for an arm. While the arm typically has higher possible precision, its main advantage of rotational movement and6 Degrees of Freedom would be lost on our device. We believe a Gantry system, as described as a “Core-ZY” in this project, offered to be a much simpler solution. By treating the front of the oscope like a 2.5 Dimensional surface, we can easily and accurately move our toolhead to manipulate and work with whatever devices Tektronix puts inside. Programming this device is also much easier than an arm, which would need to use inverse kinematics something none of our team has experience with.

### 4.1.4 Interface Validation

**MTR_CNTRL_STPPR_MTRS_COMM: Input**

### 4.1.5. Block Testing Process

Testing Physical Dimensions of the Device

1. Measure the exterior dimensions of the device and should be no larger than 20in depth x 22in height x 34in width.
2. Measure the interior dimensions of the device and should be at minimum 7.5in depth x 12in height x 17.5in width.

Testing for the input interface **STPR_MTRS_MCHNCL_STRCTR_MECH (X, Y, Z, Toolhead)**

1. Using a fish scale (Lbs. Oz.), attach the device to the lower beam of the gantry.
2. Pull with the device in hand from the front of the device to the rear.
3. The Maximum force required should not exceed 20Oz.

Testing for the output interface **MCHNCL_STRCTR_INTRRPTS_ENVIN**

1. Wire each axis limit switch
2. Connect a DMM to the ends of the limit switch, and set in continuity mode.
    a. When the switch is not pressed the DMM should report continuity.
    b. When the switch is pressed the DMM should report a break in continuity.
3. Repeat for each of the X, Y and Z axes.

### 4.1.6. References and File Links

#### 4.1.6.1. References (IEEE)

#### 4.1.6.2. File Links

### 4.1.7. Revision Table

| | |
|---|---|
| Ryan Christensen 04/22/2022 | Added Section 4.1 |

## 4.2. Stepper Drivers - Ryan Christensen

### 4.2.1. Block Overview

The stepper driver works in conjunction with the Mechanical Structure. The mechanical structure of the previous block provides the frame with physical structure while the Stepper Drivers give the frame motion, allowing the toolhead to move, and moving the linear rails along each of the axes. This can be referred to as a “Core-ZY” system, where the ZY Axis are controlled by two motors operating in parallel while the x-axis is controlled by a single motor. The X-axis uses a direct driven belt system to move the gantry either closer or further away from the oscilloscope.

### 4.2.2. Block Design

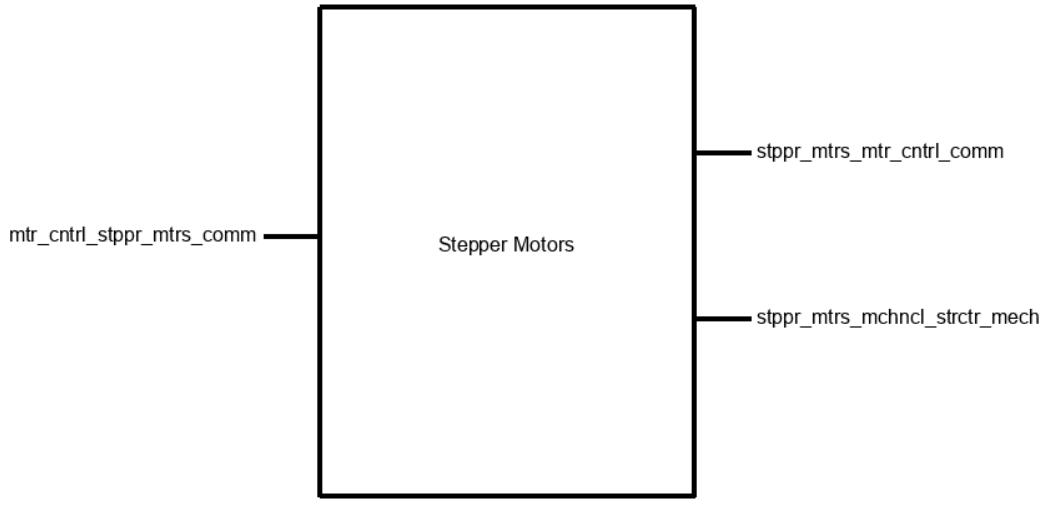



Figure 1. Black Box and Block Interface

### 4.2.3. Block General Validation

This block will give the system motion, allowing the mechanical structure to interface and operate the oscilloscope. Each of the stepper motors will operate along a belt driven system,

moving each of the axes in its necessary direction. On the circuit board we will be using the DRV8825 breakout board from pololu [1] to translate the direction and steps into the current and voltage along the motor wires. Each axis has its individual driver which can handle up to 1.5Amps continuous with 2.2Amps peak [1].

### 4.2.4 Interface Validation

**STPPR_MTRS_MCHNL_STRCTR_MECH: Output**

| | | |
|---|---|---|
| Minimum Linear Speed: 1in/Sec | The system needs to respond to an input in a timely manner, while retaining precision. Using a controlled speed like this helps to avoid unnecessary Jerk or Acceleration which may cause missed steps. | The default setting for GRBLHal was about 6in/sec. By staying at the default we give ourselves headroom of 4-5in/second in case reliability or precision becomes an issue. |
| Minimum Precision +/- .5in | The system will be working with | Using a direct belt system we can directly control the distance traveled by the motor through the number of rotations. A single step is equal to about .15mm of travel in the linear direction. Well below the .5in minimum. |
| Minimum Force Output; 20oz | The motor needs to be strong enough to move the device. | Previous testing has shown that the gantry required 6-10ozs to pull the gantry back and forth along the X axis. |

**MTR_CNTRL_STPPR_MTRS_COMM: Input**

| | | |
|---|---|---|
| Protocol: Step-Direction | The stepper driver chosen uses step and direction to interface with it [2]. | GRBL HAL[1] sends the step and direction signals given the G code where the output pins are connected to the stepper driver. |
| Logic Level: 3.3V, 0V | The stepper driver uses 3.3V and 0V for the direction [2]. The output pin from the µController does not provide 3.3V exactly so it should fall around it. | The stepper driver will need 3v3 for going one direction and 0 going the other. The µController outputs 3v3 for logic high from the datasheet. |

| | | |
|---|---|---|
| Messages: Direction to spin motor - Step or Hold motor direction | There will be 4 stepper motors and stepper drivers and each will be routed to a specific pin on the µController. One motor for each axis and one for the toolhead. The step will change, increase or stay zero to tell the stepper to move forward or stop. | Step and direction pins for each stepper are mapped for different axes in grbl HAL[1]. The µController produces a voltage on the step pin depending on the speed. If there is 0 voltage the motors will stop. |

### 4.2.5. Block Testing Process

1. Testing **stppr_mtrs_mchncl_strctr_mech**
    a. Minimum Precision +/- .5in
        i. Each axis will start with it's zero position.
        ii. Send the proper GCode to move the device to the midpoint of its axis. (I.E. 150mm on the X Axis)
        iii. The carriage of the axis should be within .5in of the mid-point.
    b. Minimum linear Speed: 1in/Sec
        i. Each Axis will start with it's zero position
        ii. Send the proper GCode to move the device from its zero to it's maximum
        iii. The carriage of the axis should reach its maximum distance within the amount of time to reach the rate of 1in per second. (I.E 15in should be 15seconds.)
    c. Motor Force Test
        i. Each Axis will be tested to confirm that it can pull more than 20 Oz.
        ii. Attach a Hanging Scale to the Axis to test, applying resistance manually
        iii. Move the axis in one direction, measuring that the motor can provide more than 20 Oz of force.
2. Testing **stppr_mtrs_mtr_cntrl_comm**
    a. Message Status
        i. Set the Sleep Pin low to turn the driver off
        ii. Change the value of step and direction of the corresponding motor
        iii. The motor should not respond to any change in it's Step or Direction.
    b. Vmin: 0V - Motor is Sleep
        i.
    c. Nominal: 3.3V - Motor is Active

3. Testing **mtr_cntrl_stppr_mtrs_comm**
    a. Logic-Level: 3.3V
        i. The logic level needs to be verified to be 3v3

   ii. Verify that 3.V Volts is being supplied to the Vin of the Motor Drivers
   iii. Verify during operation that Step and Direction inputs do not exceed 3.3V
b. Messages: Direction to spin motor - Step or Hold motor direction
   i. Using a DMM, Validate that when the Direction Pin is 3.3V. That the corresponding motor spins in direction 1
   ii. Using a DMM, Validate that when the Direction Pin is 0V. That the corresponding motor spins in direction 2
   iii. Simulating a step, Validate that when given one step, the motor moves 1.8* in rotation.
c. Protocol: Step-Direction
   i. Using a DMM, validate the Direction pin only has 3.3V or 0V during operation.
   ii. Using a DMM, validate the Step pin should only have 3.3V when moving or 0V when stationary.

### 4.2.6. References and File Links

#### 4.2.6.1. References (IEEE)

[1] DRV8825 Stepper Motor Driver Carrier
https://www.pololu.com/product/2133
[2] Stepperonline NEMA17 17HS15-1504S-X1 Full Datasheet
https://www.omc-stepperonline.com/download/17HS15-1504S-X1.pdf

#### 4.2.6.2. File Links

### 4.2.7. Revision Table

| | |
|---|---|
| 4/22/2022 | Ryan Christensen: Added Section 4.2 |

## 4.3. Interrupts - Kevin Ho

### 4.3.1. Block Overview

This block stops the system during runtime if the system reaches the bounds of the mechanical structure or before the tool head can push into the oscilloscope. This block contains 6 limit switches attached to the ends of the 3 axis of the system. The switches will be connected to the motor controller. When the switches open the signal will be sent to stop the axis from moving.

## 4.3.2. Block Design

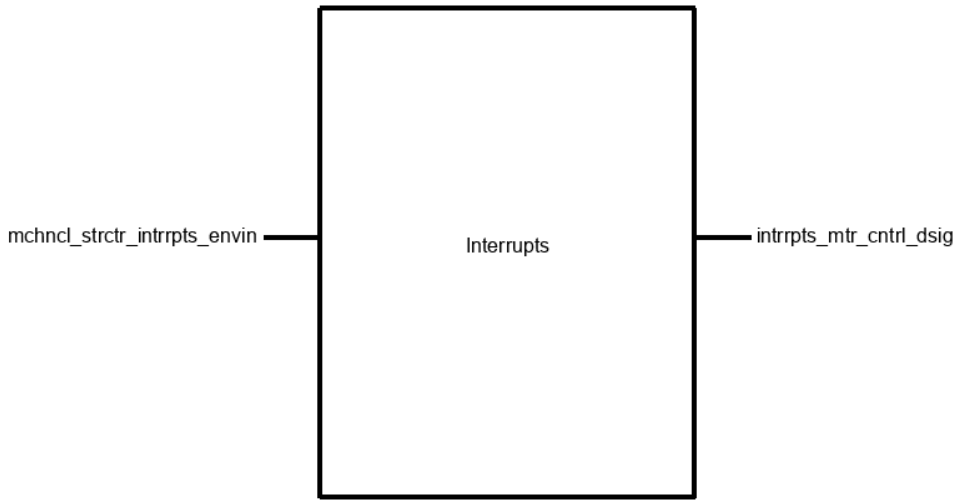



Figure 1: Black Box Diagram

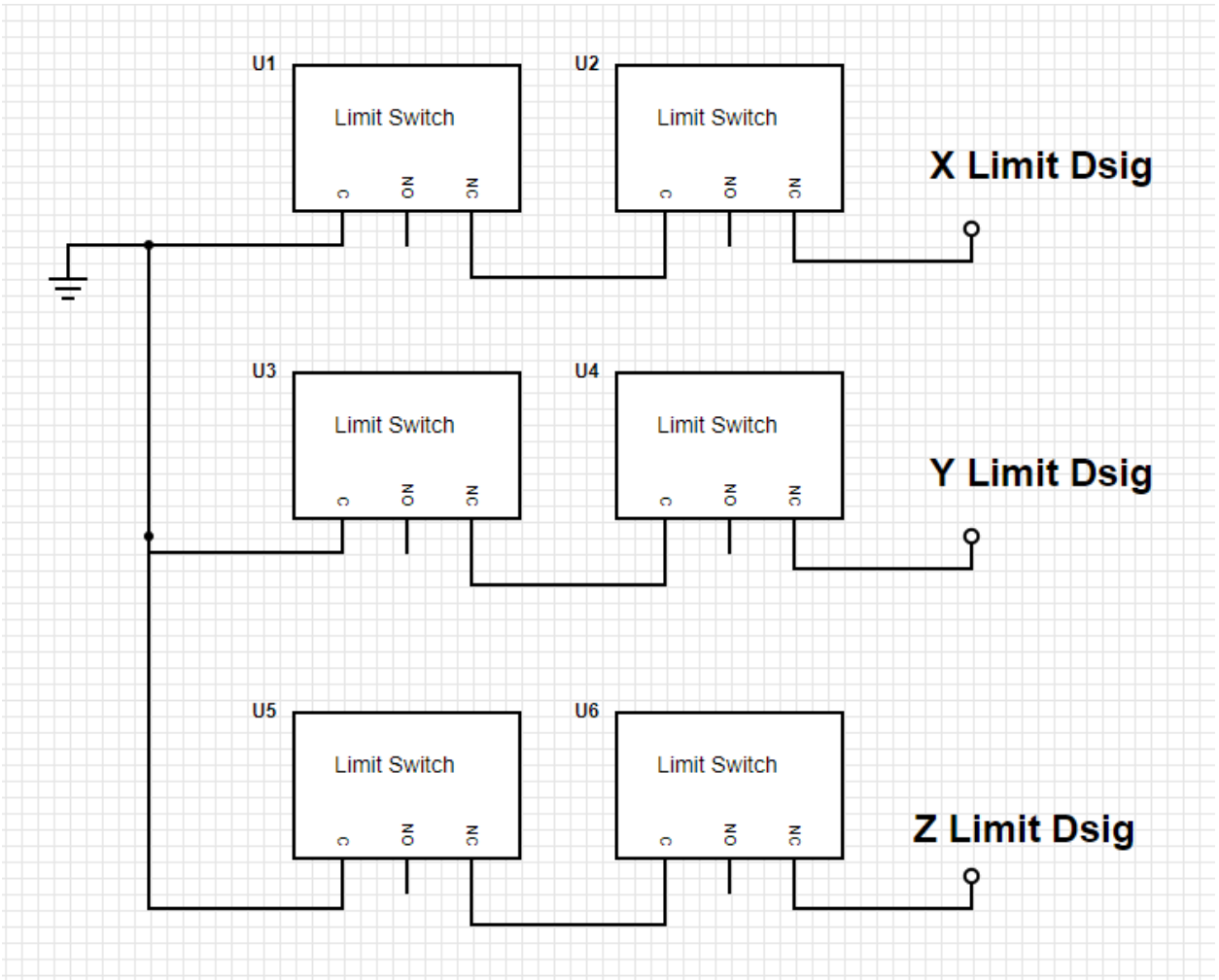



Figure 2: Connection Schematic

Pseudo Code

```
Set digital pin to input with internal pull up enabled
Loop{
        Read in the digital pin
        if(digital pin)
        Serial print(Limit Hit)
}
```

## 4.1.3. Block General Validation

This design works because if a limit switch is hit, it will produce a digital signal of 1 to the microcontroller. To produce the 5V-4.7V the internal pull up resistor will be on. The blackpill will step down to a 3.3V pull up if there is more load on the microcontroller [1]. If any limit switch is hit the microcontroller should stop the whole system in case more boundaries are pushed unless the device is homing. The limit switches should be available with a lot of options of manufacturers. These connectors will make sure that the GND will go to the C and the output will go into the digital pin. The limit switches will be attached to both sides of each axis on the mechanical structure.

Two limit switches are wired together for each axis shown in Figure 2. This will decrease the amount of pins needed for the limit switches. The limit switches will be wired normally closed through ground in case a switch is detached by accident and the machine will stop.

This block will make sure that the device won't push over the oscilloscope when the system is operating. This will help prevent any damage done from the system to the oscilloscope. The limit switches will also be used for homing in case the device turns on not in the home position.

## 4.3.4. Block Interface Validation

**INTRRPS_MTR_CTRL_DSIG**

| **Interface Property** | **Why is this interface this value?** | **Why do you know that your design details <u>for this block</u> above meet or exceed each property?** |
|---|---|---|
| Protocol: Digital Input | Each pair of limit switches will be hooked up to a digital pin on the microcontroller | The switches will produce a logic level high or low which a digital pin can take in. |

| | | |
|---|---|---|
| Logic Level: 5-4.7V, 0.3-0.0V | The signal that is needed for the microcontroller is 5V for logic 1. | The common port on the limit switch will be connected to ground. The internal pull up resistor will be on to produce a 5-4.7B signal when the switch is hit. The switches are daisy chained in a way where ground will be going through the switches unless a switch is hit. |
| Data: Limit Switch, depending on which axis switch is activated. | Since there will be more than one limit switch the data of each limit switch will be sent to the microcontroller. | The motor controller will be able to differentiate which limit switch is hit though digital input by knowing which way the tool head is moving. |

**MCHNCL_STRCTR_INTRRPTS_ENVIN (X, Y, Z, Toolhead)**

| | | |
|---|---|---|
| Description: Axis limit switch, used to home device and alarm status for Motor Controller | The limit switches will be used to check if the system is at the boundaries where we can use that information to alarm the system and home it. | The limit switches will be secured at the boundaries of the system. When the switch is closed the signal will go to the microcontroller to either stop the machine or home it. |
| Logic Level:0.3-0.0V, 5-4.7V | The limit switch will produce a signal logic high or low, 5V or 0V, depending if the structure hits the limit switch. | The common port on the limit switch will be connected to ground. The internal pull up resistor will be on to produce a 5-4.7B signal when the switch is hit. The switches are daisy chained in a way where ground will be going through the switches unless a switch is hit. |
| Data: CLOSED in normal state, OPEN when limit switch is active | The limit switch should be active when there is no physical contact on the switch. When the switch is pushed down the normal state should occur to produce the 5V signal. | The limit switch can be hooked up to be normally closed. This will create the needed interaction. |

## 4.3.5. Block Testing Process

1. Wire up a pair limit switches as shown above.
    a. To save resources only one pair is needed for testing
2. Connect the output of the limit switch to the digital pin.
3. Flash the testing code on to the arduino (the microcontroller used for testing)
4. Open the serial monitor to see the logic levels

    a. If the switch is pressed, a message will appear.
5. Measure the voltage at the pin with a DMM
    a. When the switch is not pressed the DMM should read 0.3V-0V.
    b. When the switch is pressed the DMM should read 5V-4.7V.
6. Repeat for the different digital pins.

### 4.3.6. References and File Links

#### 4.3.6.1. References (IEEE)

#### 4.3.6.2. File Links

[1] STM32 Datasheet https://www.st.com/resource/en/datasheet/stm32f411ce.pdf

### 4.3.7. Revision Table

| | |
|---|---|
| 05/06/2022 | Kevin: Added section 4.3. |
| 01/21/2022 | Kevin: Revised general description |
| 01/20/2022 | Kevin: Revision of Interface Definition/testing process/schematic |
| 01/7/2022 | Kevin: Rough Draft of interrupt block validation |

## 4.4. Motor Control - Kevin Ho

### 4.4.1. Block Overview

The motor control block receives data from the Raspberry Pi. The data received will be in gcode. It will use the data with GRBL HAL to send data to the motor drivers within the block to move the stepper motors to a xyz coordinate. The motor driver will also receive updates of the status of the motors through a digital signal. This signal is then passed on to the raspberry pi to update the user interface. The signal to is sent through serial between the raspberry pi and microcontroller. This will be used to keep track of the state of the motors. The interrupts will send a digital signal to the motor controller. This signal will be from limit switches so the motor controller will know the limits of the mechanical structure.

## 4.4.2. Block Design

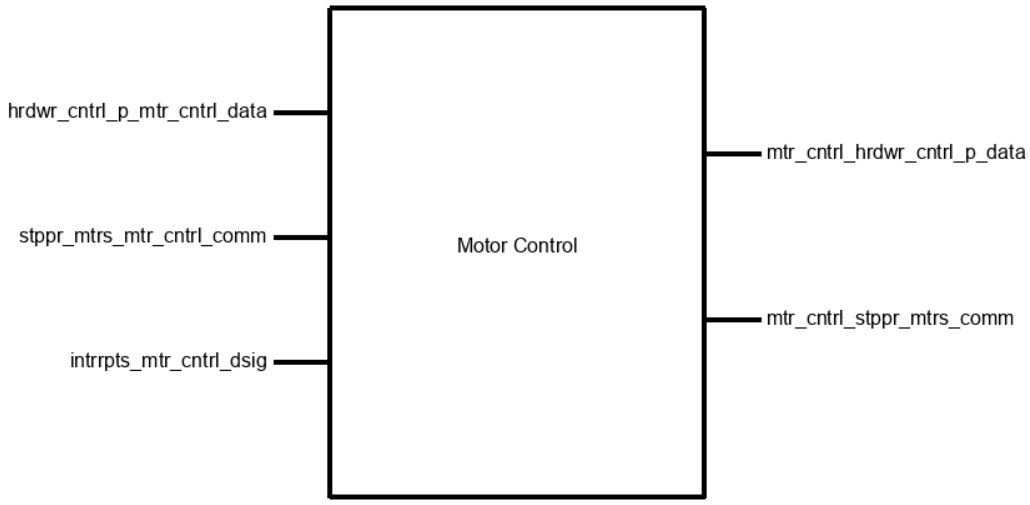



Figure 1: Black Box Diagram

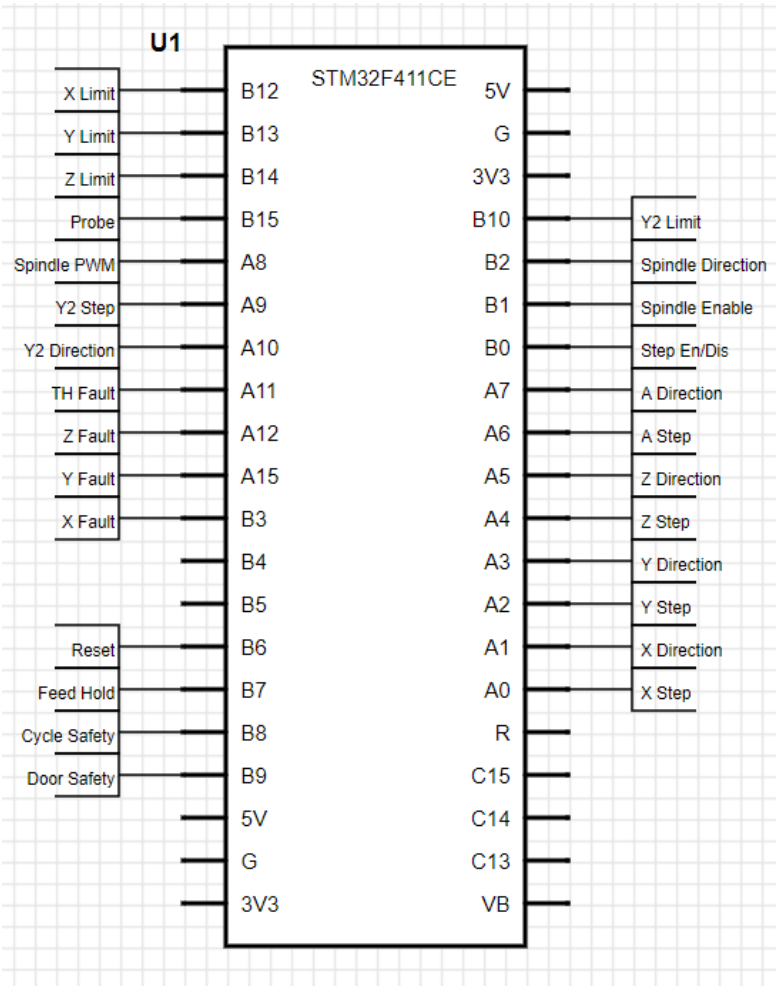



Figure 2: How pins are assigned for the microcontroller

### 4.4.3. Block General Validation

The STM32F411CE integrated circuit is sold out. So the plan is to use the development board, the Black Pill, instead. The development board was found using Adafruit. There are two options in this case where one the development board will be put on to the pcb, or the microcontroller from the development board can be harvested and put on a custom pcb. In case both options fall through, a backup microcontroller is planned to take its place, the RP2040. The options for the STM32 and the RP2040 were decided from the support that grblHAL had. The STM32 was chosen over the RP2040 because of the support that grblHAL had for the STM32 over the RP2040.

This block will work with the firmware grblHAL [1]. grblHAL is a g code decoder for a wide variety of microcontrollers. It is based on the g code decoder grbl which is for arduino microcontrollers. grblHAL is open sourced and is maintained well by active contributors. grblHAL enabled the group to add more reliable functionalities given the timeline of the project. The g code will be communicated through serial communication. Also any errors or status updates would also be communicated through the same serial communication. The serial connection is through a USB-A to USB-C cable which provides power for the blackpill.

The blackpill or microcontroller will be used to take in g code and send the correct signals to move the tool head and the stepper motor drivers. The blackpill will also take in the interrupt and fault signals in case any limits or high current draw happens. The connections for each stepper driver will be the direction and step pins. These connections will be a digital signal with either logic level high or low.

### 4.4.4. Block Interface Validation

**MTR_CNTRL_HRDWR_CNTRL_P_DATA (Motor Controller)**

| 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? |
|---|---|---|
| Protocol:USB | The connection between the raspberry pi and µController is serial. | The raspberry pi can communicate through usb using the terminal and the black pill or the STM32F4 can as well from the datasheet section 3.27. |
| Other:Data | Grbl [1] will accept a | When using the serial monitor the µController |

| | | |
|---|---|---|
| Character Arrays From and to the µController to the Raspberry Pi 4 | string or a character array and decode that for g code and other commands. | can receive arrays and send arrays through serial. |
| Other:Send one array of character containing current states | Grbl can receive a command that reports the current state of the process and gives feedback that commands are received [1]. | Grbl HAL sends updates through serial or usb [1]. These messages are put into one array. Grbl HAL will send feedback messages to acknowledge a command is sent and will send its status if asked for. |

**HRDWR_CNTRL_P_MTR_CNTRL_DATA**

| | | |
|---|---|---|
| Protocol:USB | The connection between the raspberry pi and µController is serial. | The raspberry pi can communicate through usb using the terminal and the black pill or the STM32F4 can as well from the datasheet section 3.27. |
| Other: Data Character Arrays | Grbl [1] will accept a string or a character array and decode that for g code and other commands. | When using the serial monitor the µController can receive arrays and send arrays |
| Other:Sends one GCode command at a time | Grbl [1] can decode one g code per command. | GRBL HAL[1] can receive one GCode and send the right signal to the drivers. |

**MTR_CNTRL_STPPR_MTRS_COMM**

| | | |
|---|---|---|
| Protocol: Step-Direction | The stepper driver chosen uses step and direction to interface with it [2]. | GRBL HAL[1] sends the step and direction signals given the G code where the output pins are connected to the stepper driver. |
| Logic Level: 3.3V, 0V | The stepper driver uses 3.3V and 0V for the direction [2]. The output pin from the µController does not provide 3.3V exactly so it should fall around it. | The stepper driver will need 3v3 for going one direction and 0 going the other. The µController outputs 3v3 for logic high from the datasheet. |
| Messages: Direction to spin motor - Step or Hold motor direction | There will be 4 stepper motors and stepper drivers and each will be routed to a specific pin on the µController. One motor for each axis and | Step and direction pins for each stepper are mapped for different axes in grbl HAL[1]. The µController produces a voltage on the step pin depending on the speed. If there is 0 voltage the motors will stop. |

| | | |
|---|---|---|
| | one for the toolhead. The step will change, increase or stay zero to tell the stepper to move forward or stop. | |

**STPPR_MTRS_MTR_CNTRL_COMM**

| | | |
|---|---|---|
| Nominal: 3.3V - Motor is Active | The logic pin ranges from 5.25V-2.5V which is tied to the fault pin [2]. | The pin will be connected to a gpio pin on the µController. There the microcontroller can detect whether the fault pin is high or low. |
| Vmin: 0V - Motor is Sleep | The fault pin is tied to the sleep pin [2]. Both the pins are active low so if the pin is low the motors are at fault. | The pin will be connected to a gpio pin on the µController. There the microcontroller can detect whether the fault pin is high or low. |
| Messages: Status | If this pin is pulled low the µController will know that the steppers faulted and will stop the command. | The motor drivers will produce the signal when the motors are pulling too much current. This will be done through firmware and alteration of grblHAL. |

**INTRRPS_MTR_CTRL_DSIG**

| | | |
|---|---|---|
| Protocol: Digital Input | Each pair of limit switches will be hooked up to a digital pin on the microcontroller | The switches will produce a logic level high or low which a digital pin can take in. |
| Logic Level: 5-4.7V, 0.3-0.0V | The signal that is needed for the microcontroller is 5V for logic 1. | The common port on the limit switch will be connected to ground. The internal pull up resistor will be on to produce a 5-4.7B signal when the switch is hit. The switches are daisy chained in a way where ground will be going through the switches unless a switch is hit. |
| Data: Limit Switch, depending on which axis switch is activated. | Since there will be more than one limit switch the data of each limit switch will be sent to the microcontroller. | The motor controller will be able to differentiate which limit switch is hit though digital input by knowing which way the tool head is moving. |

### 4.4.5. Block Testing Process

1. Flash in the GRBL HAL to the black pill.
2. Ground the limit pins to simulate the limit switches.
3. Put 3.3V to the fault pins to simulate the stepper driver connections.
4. Connect to the black pill using the arduino serial monitor.
    a. This will act as the raspberry pi usb communication.
    b. Make sure that the settings are set
        i. Send in the serial monitor $14 = 1
            1. This disables the door lock and cycle safety
        ii. Send in the serial monitor $X
            1. This unlocks the machine
        iii. Send in the serial monitor $21 = 1
            1. This enables hard limit
5. Send the g code g0 x30 to the black pill through the serial monitor
6. Using a DMM measure from the X direction and ground. (the X step voltage depends on speed)
    a. Should be 0V
7. Using a DMM measure from the X step and ground. (the X step voltage depends on speed)
    a. Should be around .6V
8. Send the g code g0 x0 to the black pill through the serial monitor
9. Using a DMM measure from the X direction and ground. (the X step voltage depends on speed)
    a. Should be 3.3-3.0V
10. Using a DMM measure from the X step and ground. (the X step voltage depends on speed)
    a. Should be around .6V
11. While the x axis is still “moving” unplug the ground to the Xlimit.
    a. Measure the voltage between the X step and ground.
        i. This should be 0V
    b. Reground the x limit pin.
    c. The microcontroller will need to be reset after the limit is hit.
        i. Connect ground to the reset pin B6 then unconnect it.
        ii. Send $X in the serial monitor.
12. Repeat steps 5-11, two more times changing the axis being tested.(change all X to Y, Z)
13. Unplug the 3.3V fault pin
    a. The microcontroller will be in an alarm state (check in serial monitor) by sending “?” in the serial monitor.
    b. The microcontroller will need to be reset after the limit is hit.
        i. Connect ground to the reset pin B6 then unconnect it.
        ii. Send $X in the serial monitor.
14. Repeat step 13 for all axes.

15. To get the status of the command send “?”
    a. In the serial monitor the status should be received by the arduino serial monitor.

## 4.4.6. References and File Links

### 4.4.6.1. References (IEEE)

[1] “grblHAL/STM32F4xx”*Github.* 18 Feb, 2022. [Online] https://github.com/grblHAL/STM32F4xx [Date Accessed 18, Feb 2022]

[2] “DRV8825 Stepper Motor Driver Carrier, High Current” *Pololu.* 18 Feb, 2022. [Online] https://www.pololu.com/product/2133 [Date Accessed 18, Feb 2022]

### 4.4.6.2. File Links

[3] STM32 Datasheet https://www.st.com/resource/en/datasheet/stm32f411ce.pdf

## 4.4.7. Revision Table

| | |
|---|---|
| 05/06/2022 | Kevin: Added to section 4.4 |
| 02/18/2022 | Kevin: Revision of block validation from peer review comments. |
| 02/01/2022 | Kevin: Rough Draft of Motor Control block validation |

# 4.5. Power Supply - Dennis Kichatov

## 4.5.1. Block Overview

The power supply block will supply all the power needed to run every single piece of equipment on the front panel project. The front panel will be able to plug into a wall socket using the AC switching power supply. This will allow the front panel to be movable or stationary and never run out of power. Since the motor drivers need 12 Volts they will be hooked up to the AC power supply directly. The smaller unit that requires less voltage will be controlled by a custom PCB that will regulate the voltage and current to a more required level.

Using 120 AC Voltage power supplied from a wall outlet. The team will use a switching power supply from StepperOnline[1] to regulate the AC voltage into usable 12V DC voltage and 8.5A current. From there the team will use custom DC to DC converters to regulate the voltage and current down and to be able to adjust the output power. The converters will need to lower the 12V into a workable 5 Volts and 3 Amps for the Raspberry Pi 4. Unfortunately the main chip for the 5 Volt converter broke, so the team will use a Raspberry Pi 4 power brick connected to USBC to send 5 Volts into the system PCB. This 5 Volt will power the 3.3 Volt chip and the rest of the PCB. The microcontroller will be powered from the Raspberry Pi 4. Coming from the power supply and straight to the motor driver is 12 Volts needed to power the stepper motors. A relay will act like a gate on the microcontroller and when there is no signal the power supply will be cut off from the microcontroller. The relay will be connected to the 3.3 Volt converter and will form a digital kill switch to turn off the motors.

## 4.5.2. Block Design

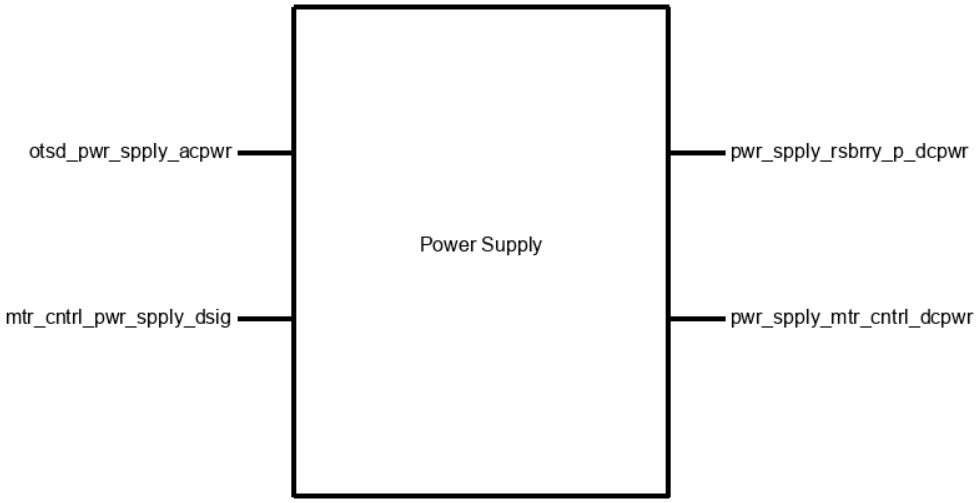



Figure 1: Top level block diagram

Figure 2: 120AC to 12DC switching power supply

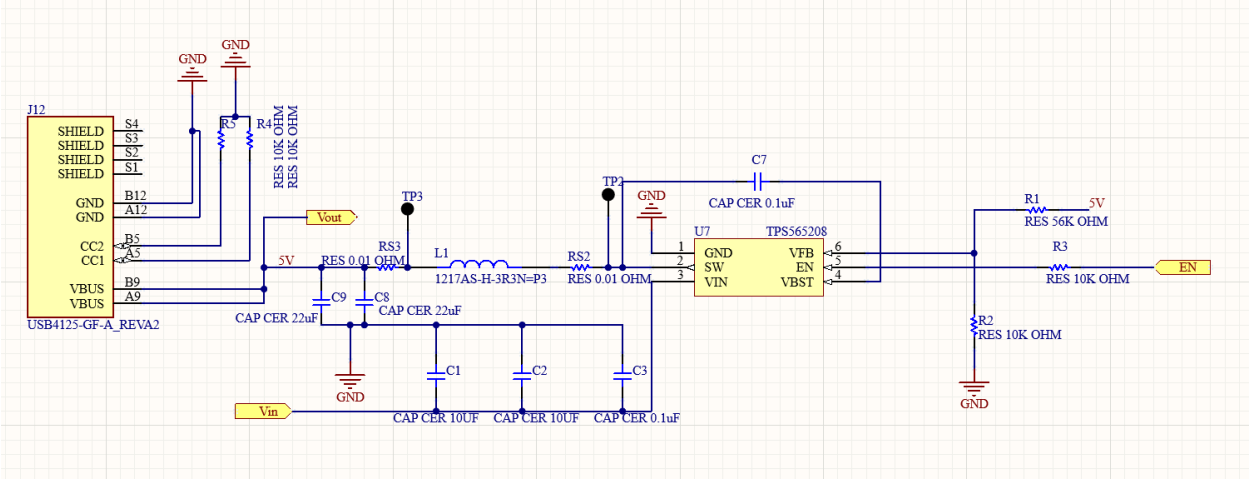



Figure 3: PCB circuit for 5 Volt regulator at 6 Amps.

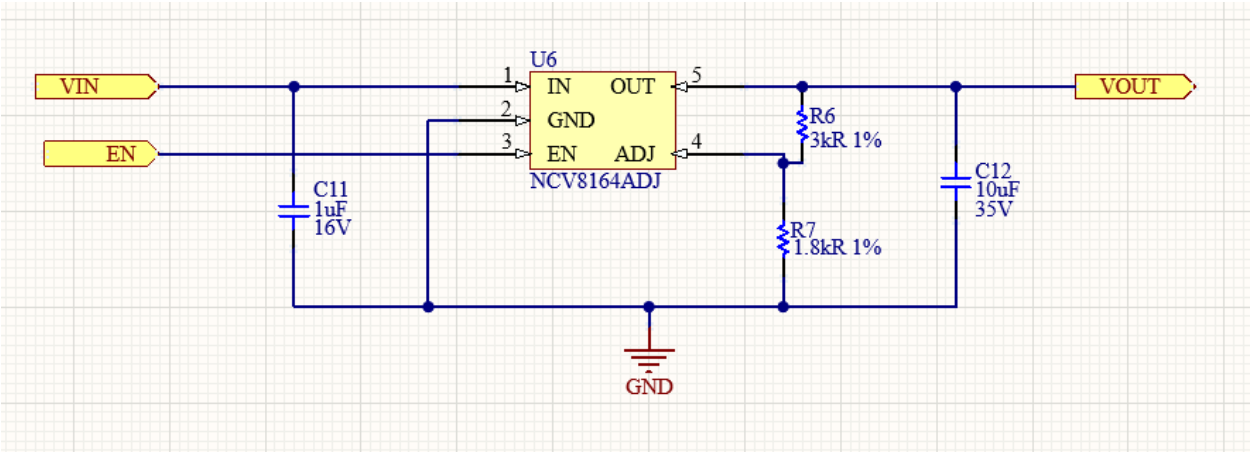



Figure 4: PCB circuit for 3.3 Volt regulator.

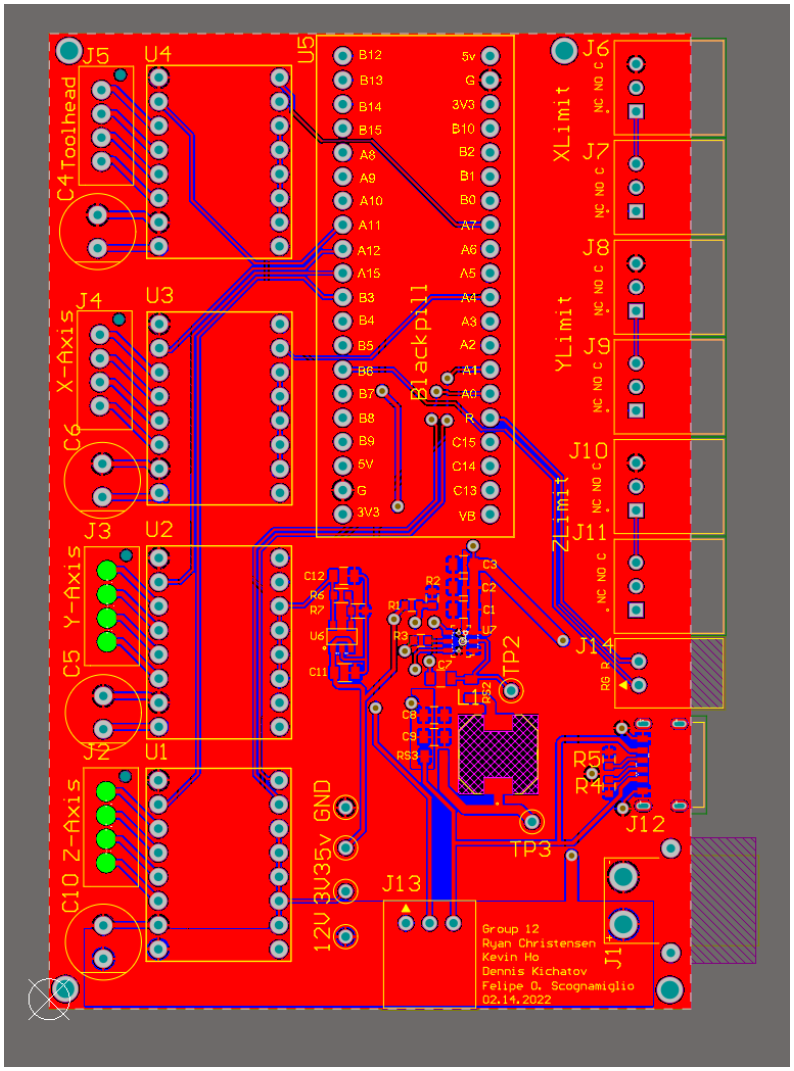



Figure 5:Top layer of system PCB. 5V reg is U7 and 3.3 reg is U6

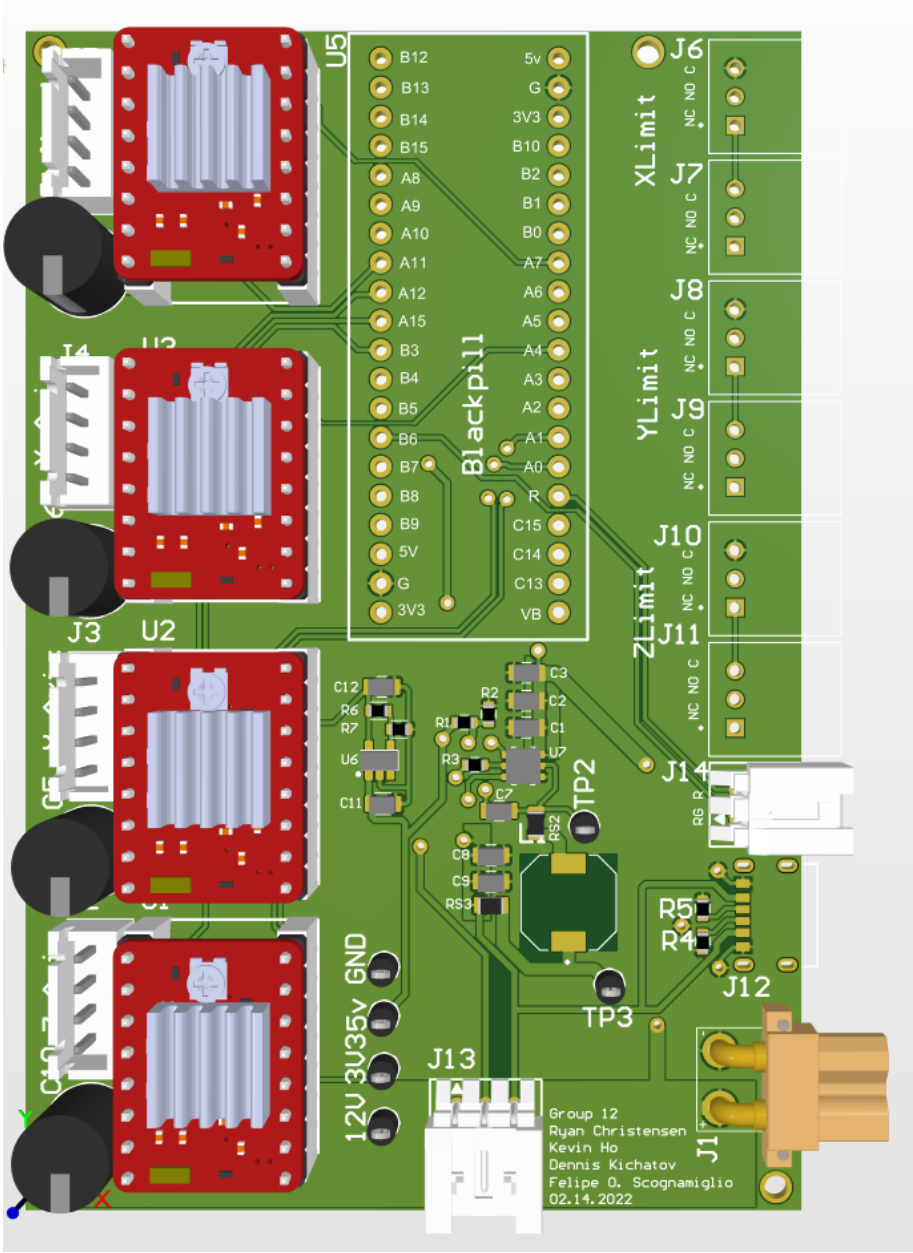



Figure 6: 3D model of System PCB and regulators.

### 4.5.3. Block General Validation

This block works for the system because the 5 Volt is based on the TPS565208DDCT Datasheet main chip which converts the 12 volts from the switching power supply to the needed 5 Volts and 6 Amps to power the Raspberry Pi 4. The resistors, capacitors and inductor components needed to build these PCBs are cheap and accessible through Resistore, Texbots on the Oregon State campus and Mouser site. The inductor was chosen to help output the 6A output. The main chip needed for the 5 volt converter is mostly out of stock, however the team managed to snag two of these chips. The majority of the cost would be paid out of pocket since going through tektronix would be too slow. The cost would be reimbursed by OSU at the end of

the winter term.The size for these PCBs will be large and attached to the enclosure. They will be able to fit inside the structure and since there are only input and output wires coming from the PCB they won't take up a lot of space. Since the main chip for the 5 Volt regulator broke and the team ran out of spares, I have decided to scrap the 5 Volt regulator and replace it with a Raspberry Pi 4 power brick connected with USBC. The brick will supply the board with 5 Volts.

The project partner Textronix, asked to make the front panel portable. The problem the team faces is by making the front panel portable the toolhead will have to calibrate every time because of this the project will not be portable. Since the project isn't moving there is no need to worry about the weight of the AC switching supply. For convenience the switching supply will be attached to the front panel. By attaching the switching power supply to the front panel it will prohibit the power supply from moving and causing problems. The two converters will also be in an enclosure that is attached to the front panel. If the Tektronix team wants to move the front panel they could.

## 4.5.4 Block Interface Validation

Otsd_pwr_spply_acpwr

| **Interface Property** | **Why is this interface of this value?** | **Why do you know that your design details for this block above meet or exceed each property?** |
|---|---|---|
| Inominal: 8.0A | This value is based on the description of the switching power supply. | From the description of the switching power supply. |
| Ipeak: 8.5A | This value is based on the description of the switching power supply. | From the description of the switching power supply. |
| Max Power:100W | This value is based on the description of the switching power supply. | From the description of the switching power supply. |
| Vnominal: 12V DC | The team needs 12 DC Volt output for the motor drivers. | From the description of the switching power supply. |
| Voltage Input: 84-264 VAC | This value is based on the description of the switching power supply. The power supply will use 120VAC from the wall socket. | From the description of the switching power supply. |

pwr_spply_rspbrry_p_dcpwr

| | | |
|---|---|---|
| Inominal: 3.0A | This is the current needed to help power the Raspberry Pi. | This is the current needed to power the Raspberry Pi. The inductor will help output 3 Amps |
| Ipeak: 3.5A | This is the peak of the current that is outputted from the 5 Volt converter. The Raspberry Pi 4 needs about 18 Watts of power. | The USB-C is able to handle around 3.0 Amps. So 3.2A will be the max current input. |
| Output Connection: USB-C | The converter will connect to the Raspberry Pi using one of these connections. | This is the required connection for the Raspberry Pi. |
| Vmax: 5.5V | This will be the maximum voltage that will be input to the Raspberry Pi. | The Raspberry Pi is able to withstand 5.5 Volts in case the voltage fluxuates. 5.5 Volt will be the max. |
| Vmin: 5V | This will be the minimum voltage requirement. Going lower than 5 Volts will dampen results. | The Raspberry Pi takes in 5 Volts of current. |

mtr_cntrl_pwr_spply_dsig

| | | |
|---|---|---|
| Imax: 100mA | This is the maximum current that flows through the relay. | This is the maximum current that flows through the relay. |
| Microcontroller connection | This is a wire that connects to the input of the microcontroller | The datasheet for the relay explains how to connect it. |
| Motor driver connection | This is a wire that connects to the input of the motor driver | The datasheet for the relay explains how to connect it. |
| Signal Pin: pulled high | The signal pin will connect to a port on the Raspberry Pi 4. When the signal is pulled high the microcontroller or motor drivers will disconnect from the power supply. | The Raspberry Pi 4 port will send a signal to switch and turn the microcontroller or motor divers “on” or “off”. |

### 4.5.5. Block Testing Process

pwr_spply_mtr_cntrl_dcpwr and pwr_spply_rspbrry_p_dcpwr

1. Create a test PCB using test points.
2. Testing points and shunt resistors will be used to test the Vmax and Vmin for the input and output.
3. A resistor will be used to test for the Inominal output.
4. Create system PCBs to test the connections for the Raspberry Pi 4.
5. The system PCB will test the relays for the microcontroller and motor drivers using a signal from the Raspberry Pi 4.

## 4.5.7. References and File Links

### 4.5.7.1. References (IEEE)

[1] Stepperonline."LRS-100-12 MEAN WELL 100W 12VDC 8.5A 115/230VAC Enclosed Switching Power Supply".[Online].Available https://www.omc-stepperonline.com/switching-power-supply/lrs-100-12-mean-well-100w-12vdc-8-5a-115-230vac-enclosed-switching-power-supply.html

### 4.5.7.2. File Links

[2] https://cdn-learn.adafruit.com/downloads/pdf/adafruit-power-relay-featherwing.pdf

[3]https://www.murata.com/~/media/webrenewal/products/inductor/chip/tokoproducts/wirewoundferritetypeforpl/m_dem8045c.ashx

[4]https://www.ti.com/lit/ds/symlink/tps565208.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&DCM=yes&distId=26

## 4.5.7. Revision Table

| | |
|---|---|
| 05/05/2022 | Worked on Power Supply Validation added section 4.5 |
| 01/21/2022 | Finialied block 1 validation for final submission |
| 01/07/2022 | Worked on block validation draft |

# 4.6. Enclosure - Dennis Kichatov

## 4.6.1. Block Overview

The enclosure for the system will keep all the circuitry and wiring together This enclosure will make sure that if the structure has to be moved all the wiring and PCB boards won't break and will be stable. This enclosure will make the wiring easy to follow and will make the project look more presentable while keeping the circuity safe. The dimensions of the enclosure are 130X350x80mm and will allow all electrical components to fit with the enclosure. There will be holes on the right and left sides to connect the enclosure to the mechanical structure. By connecting the enclosure to the structure the project as a whole can be transported with ease.

## 4.6.2. Block Design

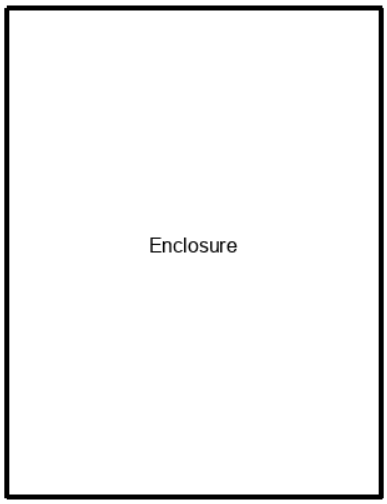



Figure 1: Top level block diagram

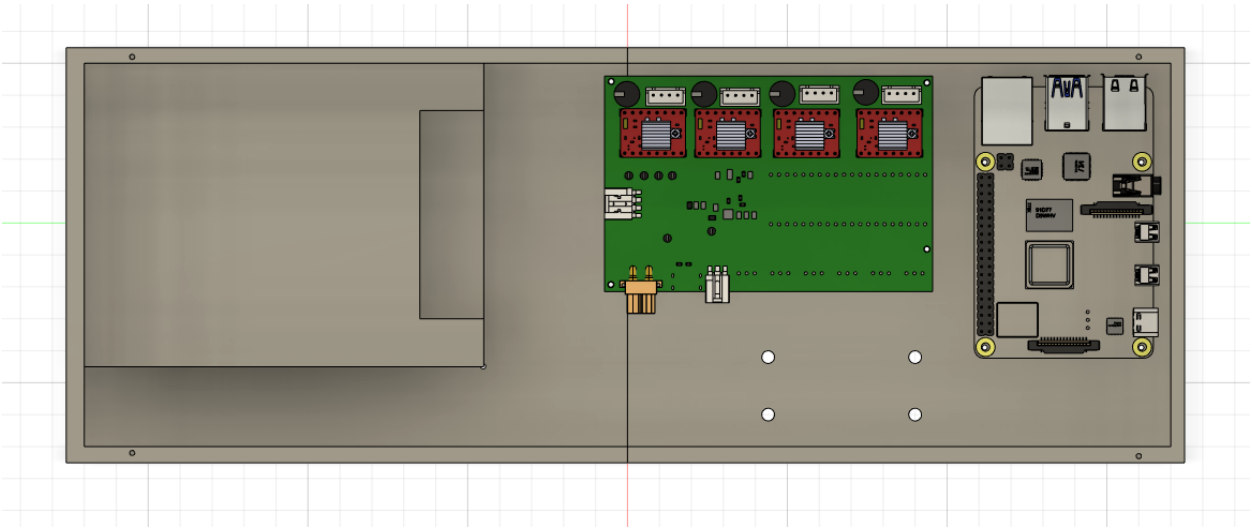


Figure 2: Enclosure with PCB, Power Supply and Raspberry Pi 4

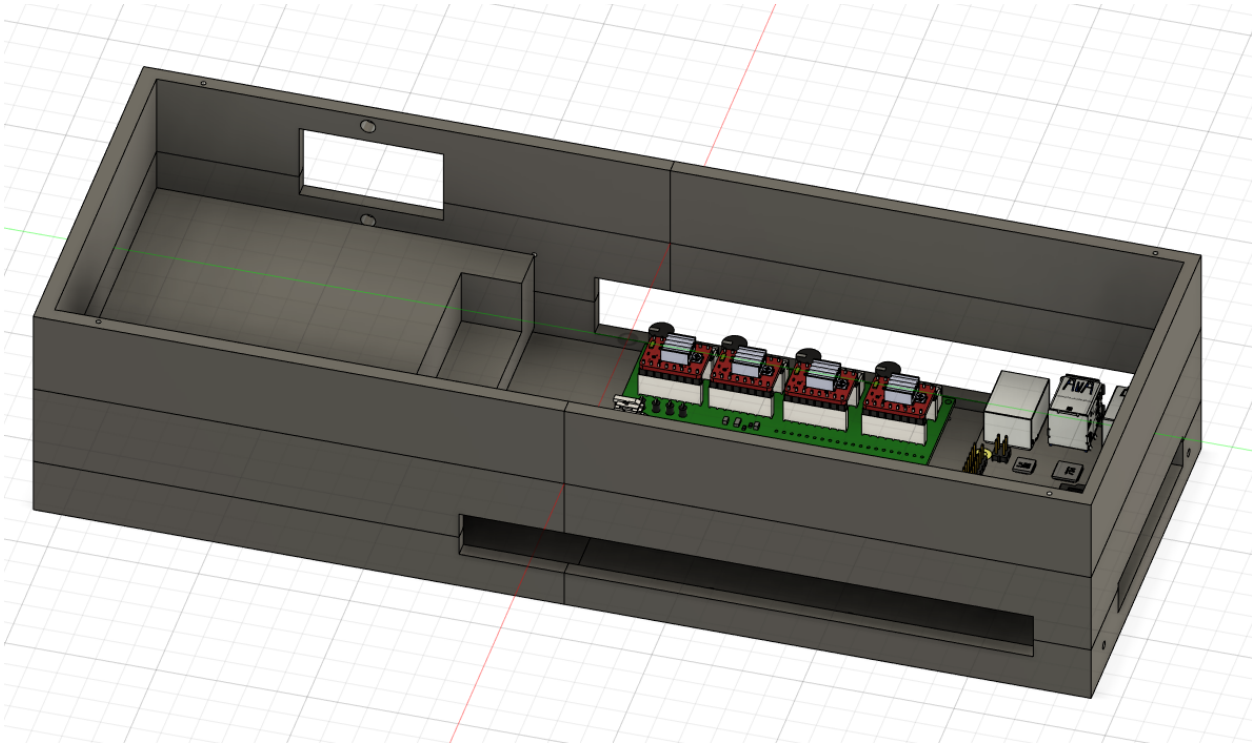


Figure 3: Enclosure side view

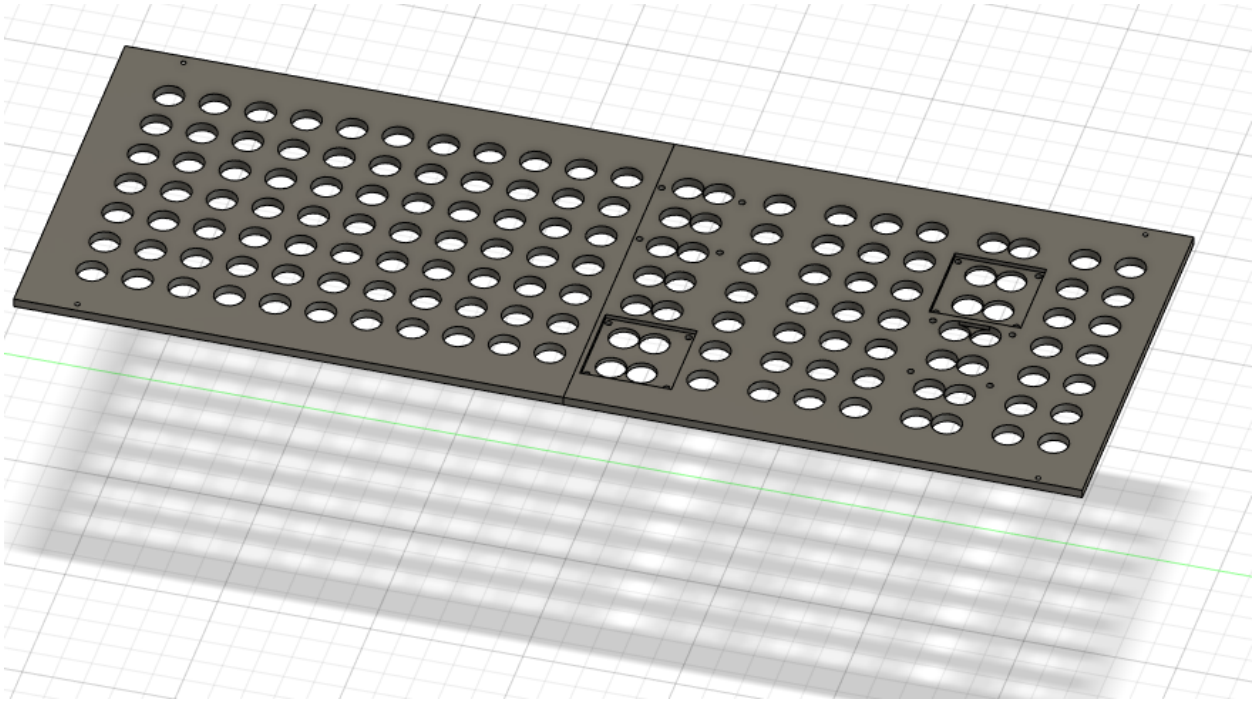


Figure 4: Lid for enclosure.

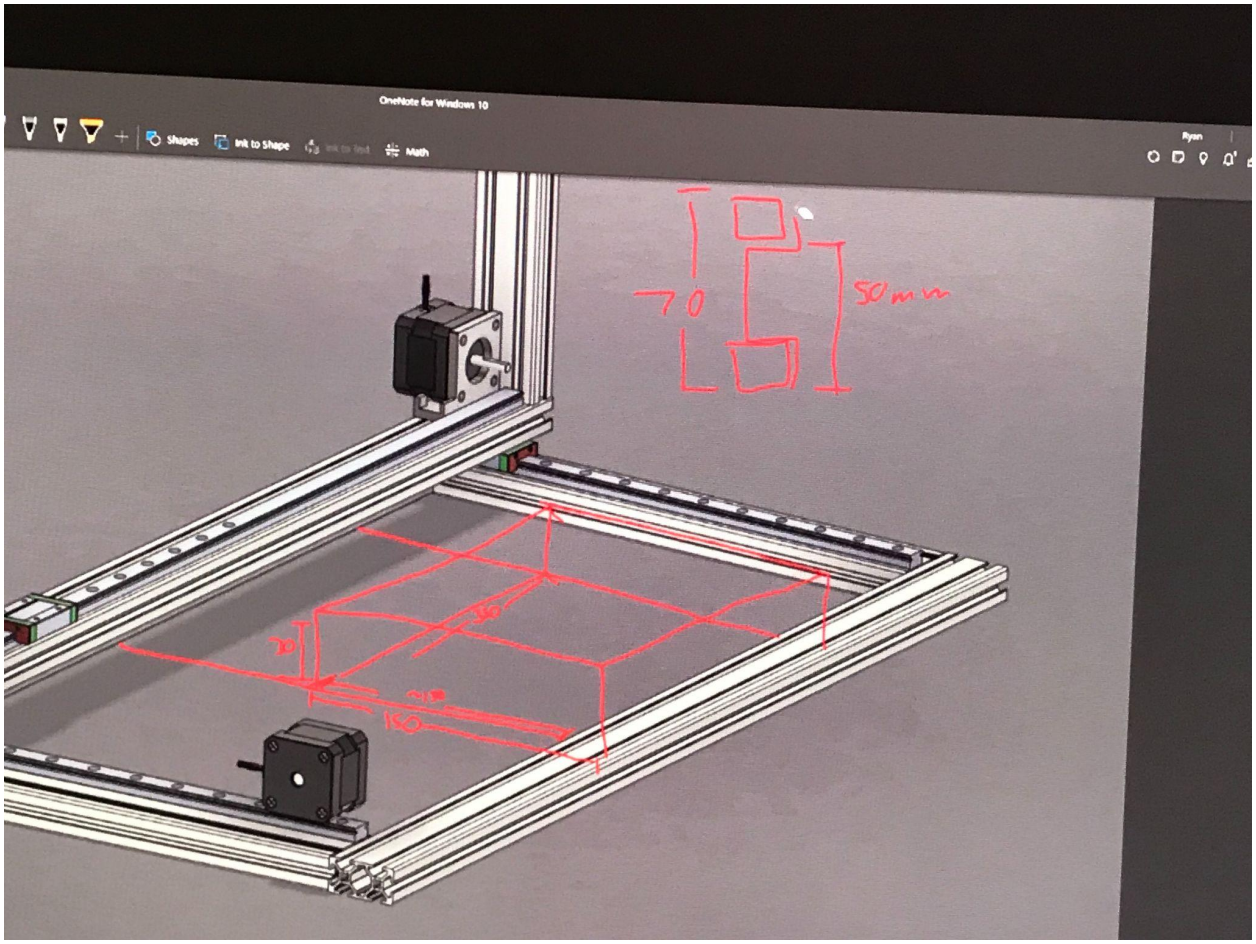



Figure 5: Enclosure with Frame

## 4.6.3. Block General Validation

This block will work for the system because the Tektronix employees hoped that the design would be portable and even if the team might not move the system we must design for movement. This means all electronics, PCBs need to be able to move for transport. The best way to ensure safety and that no movement comes to the PCBs is to enclose them in a shell. The Raspberry Pi 4, switching power supply, Black Pill and voltage regulators will be connected to a PCB to make wiring them together easier. As seen in figure 5 the enclosure will be underneath the red box frame. The Oscilloscope will sit on top of the red elevated frame and the enclosure will be underneath. Fans will be attached to the outside to cool down the Raspberry Pi 4 and to receive airflow. The enclosure will be bolted to the side of the red frame using T-nuts. The Enclosure will be created using Fusion360 and using the PCB measurements, power supply measurements and the measurements of the Raspberry Pi 4. Considering the enclosure is attached to the frame, the tool head will move up and down, left and right, forward and backward. The enclosure needs to be close enough to not have the tool head pull the wires.

## 4.6.4 Block Interface Validation

otsd_enclsr_other

| **Interface Property** | **Why is this interface of this value?** | **Why do you know that your design details <u>for this block</u> above meet or exceed each property?** |
|---|---|---|
| Dimensions | The enclosure will be large enough to fit all electronics and leave room for wiring. | Took dimensions of Raspberry Pi 4, Black Pill and switching power supply. |
| Inside | The enclosure will encompass the Raspberry Pi 4, Black Pill, step down voltage converter and stepper PCB. In a separate enclosure the switching power supply will be stored. | There will be stand offs for each component. |
| Enclosed and attached to the frame. | Both enclosures will be sealed from the top and PCBs bolted to the enclosure. | The lid will fit the enclosure and nuts and bolts will be used to attach the enclosure to the frame. |

## 4.6.5. Block Testing Process

1. Get correct measurements of Black Pill, Raspberry Pi 4 and switching power supply.
2. Create a 3D model in Fusion360.
3. Print out models.
4. Fit the models onto the Black Pill, Raspberry Pi 4 and switching power supply.
5. Sandpaper and make to fit the models.
6. Make sure all wiring fits and there is enough space for connections.
7. Use M2 bolts and nuts to attach PCBs and power supply to the enclosure.
8. Use T-Nuts to attach the enclosure to the frame.
9. Make sure the enclosure is stable and does not fall.

## 4.6.6. References and File Links

## 4.6.7. Revision Table

| | |
|---|---|
| 05/06/2022 | Finalized enclosure validation |

| | |
|---|---|
| 05/05/2022 | Worked on enclosure validation added section 4.6 |
| 02/18/2022 | Finished block validation 2 |
| 02/02/2022 | Worked on block validation draft |

# 4.7. Hardware Control API - Felipe Orrico Scognamiglio

## 4.7.1. Block Overview

The Hardware Control API receives the encoded user interface commands. The user data is then translated into GCode and put into a transmission queue. The Hardware Control API then goes through the transmission queue and sends the data packet to the motor control block. Upon receiving and completing the task, the motor control block will send a confirmation code back to the Hardware Control API announcing that it is ready to receive another data packet.

The Hardware Control API allows the abstraction of the connection between the User Interface (or another program that uses this API) and the Motor Control block that runs a version of GRBL HAL. Usually, when interfacing with GRBL the user would need to directly send GCode or other configuration commands to the microcontroller, instead, with this API, the user is able to easily set up and move the payload of the gantry system with a simple custom implementation in python.

## 4.7.2. Block Design

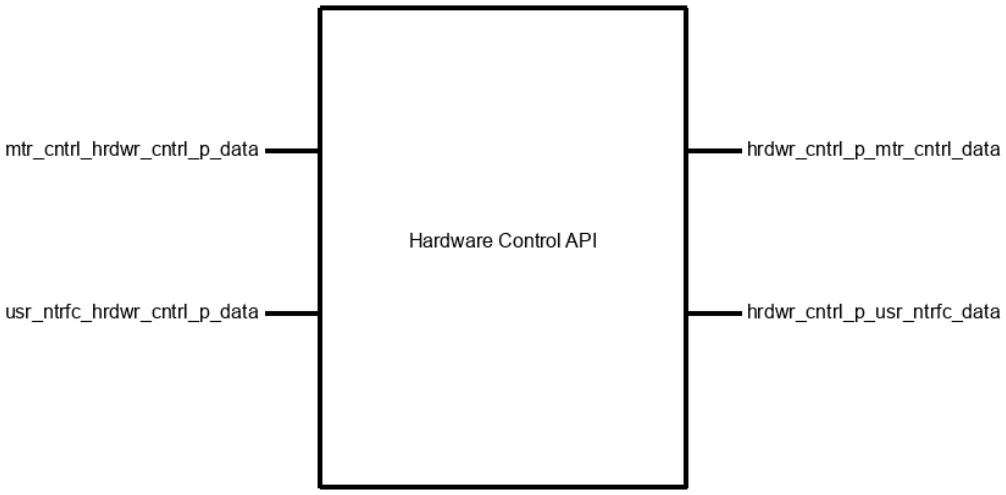



Figure 1: Motor Control API Black Box Diagram

User data is either collected by the User Interface Block or raw user input through the use of the API without the User Interface. Meaning that the user can interface with this block by either relying on the User Interface or directly calling the public methods of the API class.
This block also has the capability to send information back to the user based on user input. For example, if the user wants to know the current location of the payload, the user may request it by calling a method that interfaces with the Motor Control Block requesting positional data before relaying it back to the user with specific formatting.

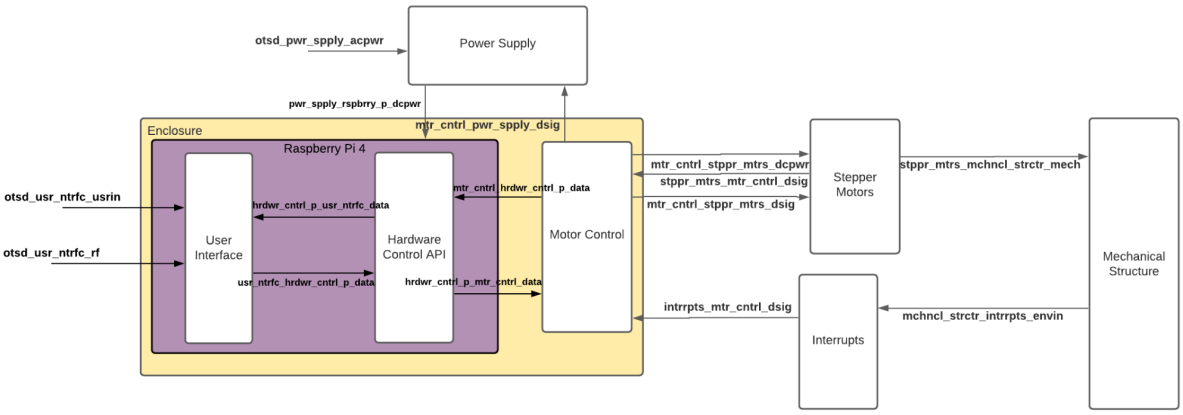



Figure 2: Hardware Control API as a part of the System.

Since this block is a layer of abstraction to the Motor Control, it has to be able to send and receive data from it. The data sent to the Motor Control Block is comprised of GCode commands or GRBL HAL-specific commands. The data received is comprised of Positional Data and movement confirmation notifications (“OK”) from GRBL HAL.

The Hardware Control API runs two separate threads. One thread is responsible for displaying live video from the Raspberry Pi camera, while the other is responsible for sending and receiving information to and from the microcontroller running grblHAL.

GCode Sender Thread (GST) - Pseudocode

- The main thread (that is running the Hardware Control API) creates an object of the GST class and requests the initialization of the GST.
- Upon receiving the list of GCode to be sent to the microcontroller, the GST appends the list to a local buffer.
- The GST then iterates through the buffer sending the code and waiting for the response from the microcontroller. At this point, the GST logs the command sent as well as the response from the microcontroller and displays it on the command line.
- Returns to Step 2 and continues.

Video Streaming Thread (VST) - Pseudocode

- The main thread (that is running the Hardware Control API) creates an object of the VST class and requests the initialization of the VST.
- At this point, the thread is running and at every frame, it updates a class-wide buffer with the current frame.
- Upon receiving the enable command, the VST uses the opencv2 library to display the video in a separate window. It continues to display video until it receives a disable command that can either be accomplished through calling a function or selecting the video window and pressing “Q”.
- Even when the video is not being displayed, the frames are being updated. Upon calling the “video_capture_frame” function from the API or pressing “S” (while the video is being displayed), the latest updated frame on the buffer will be converted to a jpeg image and saved to the current working directory of the API.

### 4.7.3. Block General Validation

The Tektronix team requested an API that would be able to control the system without the use of a GUI. By using the API as a Hardware Abstraction Layer for the GUI, it can also be said that the same API can be used without the GUI, as the only true difference between the two would be instead of receiving sanitized user input from the GUI, the API would receive raw inputs.

The main requirement of this block is being able to create a bridge between the hardware and software of the project. This API accomplishes that by allowing access to the hardware as a library for python that works with or without the GUI.

There is the possibility that the continuous run of multiple threads degrades the overall performance of the system. In that case, it may be required to implement restarting threads and blocking behavior for both the GST and the VST.

### 4.7.4. Block Interface Validation

| **Interface Property** | **Why is this interface this value?** | **Why do you know that your design details <u>for this block</u> above meet or exceed each property?** |
|---|---|---|
| **hrdwr_cntrl_p_mtr_cntrl_data : Output** | | |
| Other: Sends one GCode command at a time | The API is expected to send only GCode commands and other GRBL HAL specific commands to the microcontroller. As per the design, the API will send only one GCode command at a time while waiting for | The microcontroller receives the transmission and sends back a confirmation. If data was requested, data is sent back as well. |

| | | |
|---|---|---|
| | a response from grblHAL. | |
| Other: Data Character Arrays | The API will send strings to the microcontroller that are essentially character arrays through serial. | Both grblHAL and the API expect the use of strings to communicate with each other. The API can generate and send, through serial, messages to the microcontroller. |
| Protocol: USB | The microcontroller is connected to the raspberry pi through USB (serial) | The raspberry pi has multiple USB ports and is able to connect to the microcontroller without a problem. With the use of the pyserial library, the API is able to interact with the USB connection and send and receive commands to the microcontroller. |

**hrdwr_cntrl_p_usr_ntrfc_data : Output**

| | | |
|---|---|---|
| Messages: Hardware Confirmation and Location Data | After receiving a request, the API will return send back the requested data to the user. At this point, the data is limited to Status confirmations and Locational data. | The API is able to receive the string from the microcontroller, parse the necessary values from it, and return it to the user. |
| Other: List of character arrays for x, y, z position | The work position and machine position are reported as x, y, and z positions. | The positional data is parsed by the GST and returned to the API as a list of strings, each containing the x, y, and z coordinates. |
| Other: Character array containing machine status | The machine status is reported as “Idle”, “Busy”, among other values. All those values are strings sent by the microcontroller. | The machine status data is parsed by the GST and returned to the API as a single string. |

**mtr_cntrl_hrdwr_cntrl_p_data : Input**

| | | |
|---|---|---|
| Other: Sends one GCode command at a time | The motor controller is expected to send only confirmations and positional data to the Raspberry Pi. | The API receives the transmission and is able to process the data. If, for example, positional data is expected to be received, it will properly process and return to the |

| | | |
|---|---|---|
| | | user. |
| Other: Data Character Arrays From and to the µController to the Raspberry Pi 4 | The microcontroller will send positional data and confirmations to the API. That information is sent as a string that is essentially a character array through serial. | Both grblHAL and the API expect the use of strings to communicate with each other. The API can generate and send serial messages to the microcontroller. |
| Protocol: USB | The microcontroller is connected to the raspberry pi through USB (serial) | The raspberry pi has multiple USB ports and is able to connect to the microcontroller without a problem. With the use of the pyserial library, the API is able to interact with the USB connection and send and receive commands to the microcontroller. |

**usr_ntrfc_hrdwr_cntrl_p_data : Input**

| | | |
|---|---|---|
| Messages: Translated input from user interface to Hardware Control API | After the user interfaces with the GUI, sanitized data is sent to the API to be interpreted and transmitted to the Motor Control block. | Received data is understood and an appropriate response is triggered. If, for example, the data informs that the payload should move, relative to the current position, 3 cm on the X-axis, the API translates it to GCode and adds it to the broadcast queue. |
| Other: Receives usable user input to request video frames, take snapshots, or set text over video feed | The API is capable of sending back to the user raw video frames (usually as a NumPy array), set text over the video stream, or taking snapshots. | The API is able to request the VST to send the latest frame as a pixel array, capture that frame as a jpeg image or add text over the streaming video. |
| Other: Receives usable user input information to translate to GCode commands and add to daemon queue | The API is capable of translating raw positional data such as x, y, and z and feed rate F to Gcode commands that are queued in the GST. | The API is able to generate Jogging commands to grblHAL that accept incremental or absolute coordinates and feed rate. The command is then added to the send queue. |

### 4.7.5. Block Testing Process

1. Create a simple testing script for the API. This includes flashing grblHAL to a microcontroller and a testing python script for the API.
2. Test the API's capabilities on sending and receiving data through serial to the microcontroller
    a. Test sending GCode and waiting for confirmation before sending another command
    b. Test sending grblHAL specific commands (?, $10=0) and processing received data properly before sending to the user.
    c. Test receiving an error message and logging it.
3. Test API's capabilities on sending and receiving data to the user.
    a. This will be done using a custom python test script. The script will simulate user inputs through the GUI and make requests to the microcontroller through the API.
        i. Test if the API sends raw text from the user as expected to the microcontroller.
        ii. Test if the API sends sanitized text from the user as expected to the microcontroller by calling methods such that the gcode command is not written by the user.
        iii. Test if the API sends information received from the microcontroller back to the user as expected (Machine Location, Work Location, Machine Status).
4. If available, Test API's capabilities on loading GCode Lists as requested by the user and send proper commands to the microcontroller (considered as raw/unsanitized inputs).
5. If available, test API's capabilities on sending/displaying live video information to the user.

## 4.7.6. References and File Links

### 4.7.6.1. References (IEEE)

### 4.7.6.2. File Links

## 4.7.7. Revision Table

| | |
|---|---|
| 03/06/2022 | Felipe: Added to Section 4.7 from block validation |

# 4.8. Graphical User Interface - Felipe Orrico Scognamiglio

## 4.8.1. Block Overview

The user interface receives user input through direct interaction with the user interface The user interface interacts with the Hardware Control API by translating the user inputs into usable data that is then sent to Hardware Control API for translation and serial forwarding. The user input can be considered as raw or sanitized input based on the source.

The user is able to use preset commands by pressing a button in the User Interface, typing commands in the terminal, loading a script file, and loading a labels file.

## 4.8.2. Block Design

The user interface receives information from the user or the Hardware Control API (henceforth regarded as HCAPI). The input from the HCAPI consists of positional data, values returned from the microcontroller and other information that is available. The user input consists of keyboard input through a command-line interface available within the User interface, mouse clicks in buttons available within the User interface, Files containing positional data for buttons and knobs as well as knob height and diameter, and GCode scripts.

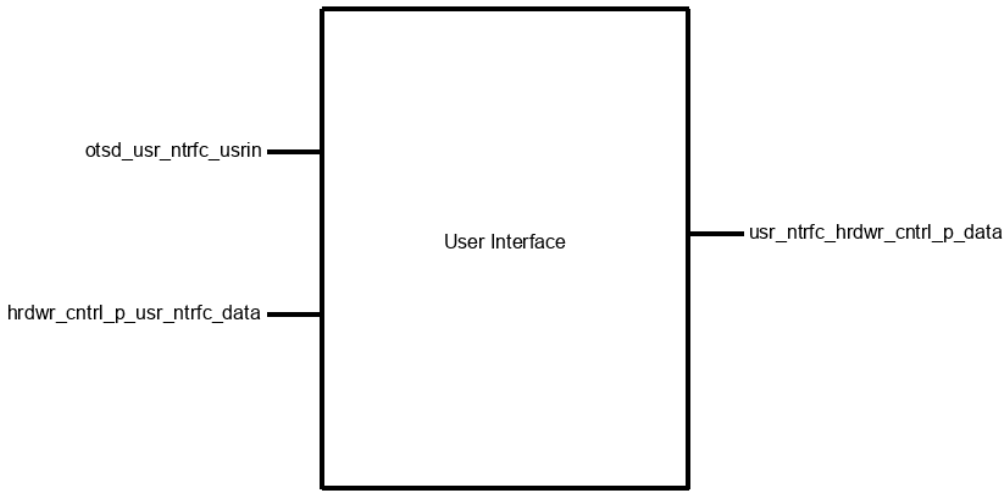



Figure 1: Black-Box Block Diagram

The user interface outputs data to the HCAPI. This data consists primarily of sanitized positional data to be sent to the microcontroller (namely x, y, and z positions, and feed rate), and unsanitized raw user input commands through a terminal within the user interface, the label files, or the script file.

The User interface runs 2 different processes. The Parent process of the UI is responsible for rendering the user interface (updating necessary values), catching interface events, and communicating with the HCAPI. The Child process is forked from the Parent process and

executes a server loop for camera controls (This had to be done due to a conflict between GTK and OpenCV2).

The GTK library handles all the events from the GUI. This happens in an inaccessible main loop, for that reason, it is not possible to run anything else besides the GUI within the same thread. In order to update values in the GUI, one needs to either be a part of the GUI thread or request the update to happen in the next render loop (from a separate thread).

1. Parent Process - Main Thread Flow
    a. Load configuration files.
    b. Load HCAPI.
    c. Fork HCAPI and Execute Video Server (go for 3 for more information on the Video Server).
    d. Create a thread to update machine position and status (go to 2 for more information).
    e. Build and show the main window for GUI.
    f. Initiate infinite render loop.
        i. Listen for events (button presses)
        ii. If the main window is closed, exit the render loop, kill the child process, and exit.
2. Parent Process - Position Update Thread Flow
    a. Request position and status from the HCAPI.
    b. Send update request to Main Thread to update the values on the GUI.
    c. Go back to ‘a’.
3. Child Process - Video Server
    a. Load Video Streaming Thread library.
    b. Initiate library frame updater thread.
    c. Enter Infinite main loop
        i. Wait for command.
        ii. Process command and call necessary methods.
        iii. Go back to ‘i’.

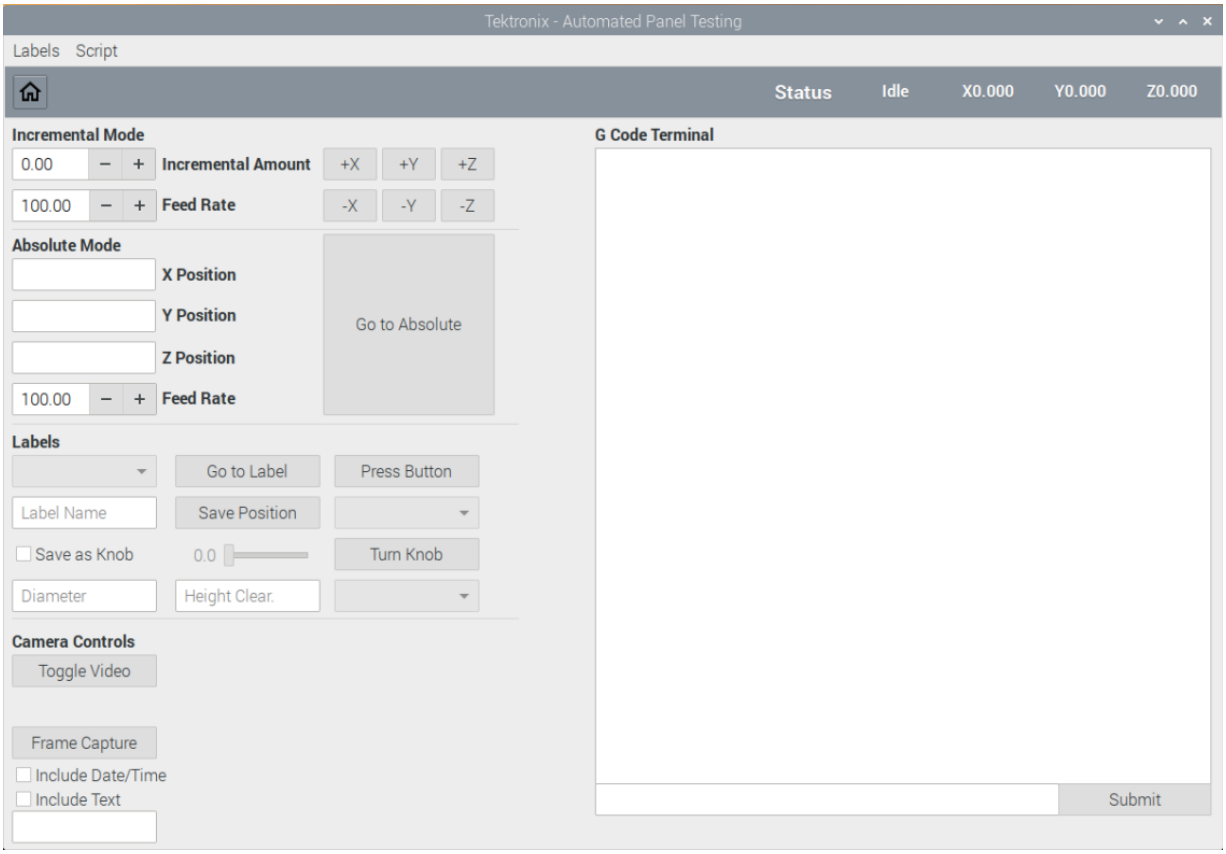



Figure 2: User Interface

The figure above (2) shows the latest design of the GUI. The top menu allows the user to interact with Labels and Scripts in the GUI. Labels can be loaded, saved, saved as, or cleared. Scripts can be loaded and executed. Whenever a script is loaded, it will remain loaded until the GUI is closed or the user loads another script.

The user is able to toggle the video window and capture the current frame. Later on, there are plans to implement the recording of the video feed for later use.

## 4.8.3. Block General Validation

This block's main function is to provide an easy-to-use interface to the user in a way that is able to translate button pushes into usable data to the HCAPI. The design will conform to this requirement by being able to demonstrate that the usability of the API is not restricted while using the User Interface.

The user interface, in its current state, is able to communicate with the API to request information from the microcontroller or carry out tasks. Currently, the main limitation of the interactions between the API and the GUI is related to the incompatibility between the Video Streamer API and the GUI library. For that reason, some changes had to be made and that made it quite difficult for the API to send video-related data to the GUI. However, the API does

not need to receive any video information, so, in practice, there were essentially no drawbacks to the changes, and validation overall will be maintained. The GUI is able to, however, through the API, send commands to the Video Streamer to request it to take screenshots, toggle the video stream, or add text over the video (used only when a preset command is executed). This allows the GUI to maintain the necessary functionality.

The user interface interface works as intended. It allows the user to load files for both scripts and labels and execute those commands. It is also able to receive typed user inputs through the terminal available inside the GUI, or button clicks through the multiple available buttons (in order to load scripts or labels the user must interact with the button menus available at the top of the GUI).

### 4.8.4. Block Interface Validation

| **Interface Property** | **Why is this interface this value?** | **Why do you know that your design details <u>for this block</u> above meet or exceed each property?** |
|---|---|---|
| **otsd_usr_ntrfc_usrin : Input** | | |
| Other: User click on button or write GCode in User Interface. | The Mouse and Keyboard are connected to the Raspberry Pi. Naturally, any GUI that is running on the Raspberry Pi is able to receive inputs from the keyboard and mouse. | The Raspberry Pi is able to read the information sent from the mouse and keyboard. The GUI should be able to receive this data whenever is selected as a window. The GUI has multiple buttons. Each button is clickable. The GUI also has a text box that will serve as a terminal that will accept written commands. |
| Other: Script File | User is able to load a script file containing a list of GCode commands to be sent to the microcontroller. | The GUI is able to load and run the script. Information about execution is added to the terminal. |
| Other: Labels File | User is able to load a file containing a list of Labels (.csv format) to be used as macros. | The GUI is able to load and interpret the list of labels and move the payload when requested. |
| **hrdwr_cntrl_p_usr_ntrfc_data : Input** | | |

| | | |
|---|---|---|
| Messages: Hardware Confirmation and Location Data | After receiving a request, the API will send back the requested data to the user. | The API is able to receive the string from the microcontroller, parse the necessary values from it, and return it to the user. |
| Other: List of character arrays for x, y, z position | The work position, and machine positions are reported as x, y, and z positions. | The positional data is parsed by the GST and returned to the API as a list of strings, each containing the x, y, and z coordinates. |
| Other: Character array containing machine status | The machine status is reported as “Idle”, “Run”, among other values. All those values are strings sent by the microcontroller. | The machine status data is parsed by the GST and returned to the API as a single string to the User. |
| **usr_ntrfc_hrdwr_cntrl_p_data : Output** | | |
| Messages: Translated input from the user interface to Hardware Control API (button pushes) | After the user interfaces with the GUI, sanitized data is sent to the API to be interpreted and transmitted to the Motor Control block. The API is capable of translating raw positional data such as x, y, and z and feed rate F to Gcode commands that are queued in the GST. | Received data is understood and an appropriate response is triggered. If, for example, the data informs that the payload should move, relative to the current position, 3 cm on the X-axis, the API translates it to GCode and adds it to the broadcast queue. |
| Other: Sends request to set text over video stream, and take frame captures | The API is capable of setting text over the video stream or taking snapshots. Text is automatic, but frame captures require the user to name the file and press a button. | The API is able to request the VST to capture the current frame as a jpeg image or add text over the streaming video. (this should be a simple function call) |
| Other: Raw GCode commands to be sent to microcontroller (input from the terminal, script, or labels) | The API is capable of translating raw positional data such as x, y, and z and feed rate F to Gcode commands (labels).<br>Terminal and Script commands are sent without checking (unsanitized). | The API is able to generate commands to grblHAL that accept incremental or absolute coordinates and feed rate (from labels).<br>The API is capable of receiving Gcode commands from the terminal or script to send to the microcontroller. |

## 4.8.5. Block Testing Process

In order to be able to verify the user interface, the interfaces that connect to the HCAPI need to be available. For that reason, within the User Interface, an object of the HCAPI will be created and serve as the interface test for this block.

Initially, to test the interface, the User Interface will call methods from the HCAPI while passing sanitized (incremental movement by pressing available buttons in the UI) and unsanitized (typing commands, loading a GCode script, or loading a labels CSV file)  user data to the API, this way, we can prove that the HCAPI can handle receiving the data from the user interface. Next, the User Interface would wait for the response back from the HCAPI regarding the information sent in the previous step. By being able to receive, interpret and display those values, we can prove that the User Interface is able to receive information from the HCAPI.

In order to test the outside user input to the HCAPI, the User Interface that is running on the raspberry pi is able to receive and interpret mouse clicks and keyboard commands translating them into usable data for the HCAPI. Keyboard commands are restricted only to raw commands to be sent to the HCAPI through the terminal present inside of the UI.

## 4.8.6. References and File Links

### 4.8.6.1. References (IEEE)

[1] GTK3 - Project Page (GUI Library)
[2] Glade - Project Page (GUI XML Builder)

### 4.8.6.2. File Links

## 4.8.7. Revision Table

| | |
|---|---|
| 03/06/2022 | Felipe: Added 4.8 from block validation |

# 5. System Verification Evidence

## 5.1. Universal Constraints

### 5.1.1. The system may not include a breadboard

The team has created a system PCB where it holds all of the blocks that need to be connected together. This reduces the amount of wires for connections and eliminates the need for a bread board. The PCB also has XT-30s and JST connectors to connect to the power supply, interrupts and stepper motors. The raspberry pi will be connected through a USB-C for power from the PCB. The microcontroller interfaces with the raspberry pi through another USB connection. This means that all the connectors can be used for interfaces between blocks and no bread boards are needed for assembly.

### 5.1.2. The final system must contain both of the following: a student designed PCB and a custom Android/PC/Cloud application

A PCB was designed to house the main electrical components. This reduced the wiring needed between blocks and made our design more compact. Three members of the team contributed to this PCB; Dennis, Kevin and Ryan. Felipe created the application that will run on the Raspberry Pi 4. This can be used from the Raspberry Pi. The application will use the API created by Felipe to send commands to the microcontroller block. This will be the main way that users will use this product.

### 5.1.3. If an enclosure is present, the contents must be ruggedly enclosed/mounted

An enclosure is designed for this project to house the PCB, Raspberry pi, relay and power supply. Mounting holes for each component has been planned out to secure them to the enclosure. The team decided to use M2 screws and standoffs to mount each component. The lid will also be screwed on so everything will be secured. The enclosure will also be attached to the mechanical structure underneath the oscilloscope to ensure that the enclosure will not move during operation of the device. Since this enclosure is designed by Dennis, it will be 3D Printed.

### 5.1.4. If present, all wire connections to PCBs and going through an enclosure (entering or leaving) must use a connector

As stated in 5.1.1 all connections from the PCB to other blocks are using connectors on the PCB. The connectors used are composed of XT-30s, JSTs, and USB connectors. These are connections already planned out on the PCB. The enclosure also accounts for the footprint of the PCB, so all connectors can connect to the PCB.

### 5.1.5. All power supplies in the system must be at least 65% efficient

The switching power supply that is used on the project is 91 percent efficient [1]. The output of the power supply is 12 volts where it is stepped down to 5 volts through a switching buck converter. This was the original plan however because of chick shortages and PCB delivery times the team decided to use a Raspberry Pi 4 power brick to supply the 5 Volts to the board.This 5v is then stepped down to 3V3 volts by a voltage regulator. This will increase the efficiency of the regulator since it does not have to heat up as much to step down the voltage.

### 5.1.6. The system may be no more than 50% built from purchased modules

Four of the eight blocks used purchased modules. The black pill, stepper motor drivers, raspberry pi and switching power supply are all purchased. The black pill and stepper motor driver daughter boards were purchased due to component shortages. The Raspberry Pi 4 was purchased to be used as an interface which acts like a PC and an application. The blocks do not include the tool head block which will be made, the designed portions of the stepper motor and power step downs. This equates to around 44% of the project being purchased.

## 5.2. The system will press a button.

### 5.2.1. Requirement

The tool head will be able to interact and activate the buttons on the front of the oscilloscope, and properly trigger the buttons without touching the other buttons on the device.

### 5.2.2. Testing Process

1. The command to push a certain button is sent by the user by selecting a saved button and pressing the designated button on the GUI to send the command.
2. The tool head will move to the correct button that is sent by the user.
3. The tool head will push the button.

### 5.2.3. Testing Evidence

5.2.3.1 Testing Evidence

## 5.3. The system turns multiple sized knobs.

### 5.3.1. Requirement

The tool head will be able to grip and turn a knob on the front of the oscilloscope, without touching the other knobs on the device.

### 5.3.2. Testing Process

1. The command to turn a knob is sent by the user by selecting a saved knob and pressing the designated button on the GUI to send the turn command.
2. The tool head will move to the correct knob that is sent by the user.
3. The tool head will grip the corresponding knob.
4. The tool head will turn the knob.

### 5.3.3. Testing Evidence

## 5.4. The system will have autonomous operation.

### 5.4.1. Requirement

Engineering Requirement: The user will use the GUI to request movement to a label location or a button press and the system will move the toolhead accordingly.

### 5.4.2. Testing Process

1. The user will send a command to move the device along the X, Y, or Z axis, using a saved label location by selecting a label location or button location and pressing the corresponding button in the GUI to send the command.
2. The user interface will interpret the commands and send the gcode.
3. The motor controller will follow the g-code received and move the mechanical frame to the designated positions.

### 5.4.3. Testing Evidence

## 5.5. The system will allow a user to manually control the device.

### 5.5.1. Requirement

The user will be able to manually control the system over a GUI and move the toolhead to its location, similar to jogging on a CNC Machine.

### 5.5.2. Testing Process

1. The user will send a command to the system to move the device along the X, Y, or Z axis, by typing the gcode into the terminal on the GUI.
2. The user interface will relay the commands the the motor control block.
3. The motor controller will follow the gcode received and move the mechanical frame to the designated positions.

### 5.5.3. Testing Evidence

5.5.3.1 Testing Evidence

## 5.6. The system will be able to toggle power to the stepper motors through the GUI.

### 5.6.1. Requirement

The user is able to toggle the relay powering the stepper motors of the mechanical structure by pressing the power button on the GUI.

### 5.6.2. Testing Process

1. The user presses the power button located on the top left portion of the GUI.
2. The system will turn ON or OFF the relay powering the stepper motors.
3. The current state of the relay can be seen updated on the GUI

### 5.6.3. Testing Evidence

## 5.7. The system will produce a video feed

### 5.7.1. Requirement

The system will collect a video feed from a device attached to the Raspberry Pi 4. The video feed will be accessible to the user in the user interface. The video feed can be recorded for future use.

### 5.7.2. Testing Process

1. Inside the User Interface, the user will click on the toggle video feed button.
2. The user starts recording the video by typing in a file name on the text field and toggling the record video button.
3. Video starts being recorded to the desired filename.
4. The user stops the video recording by toggling the video recording button.
5. Videos are saved to the filename given by the user on the user interface folder.

### 5.7.3. Testing Evidence

5.7.3.1 Testing Evidence

# 5.8. The Gantry System fits within a defined size

## 5.8.1. Requirement

The mechanical system will fit inside a 22in height x 34in width x 20in depth space. This is to ensure that Tektronix will be able to fit the device on either a shelf or a desk with easy access.

## 5.8.2. Testing Process

1. Using a measuring tape, the system width will be shown to be no greater than 34in.
2. Using a measuring tape, the system depth will be shown to be no greater than 20in.
3. Using a measuring tape, the system height will be shown to be no greater than 22in.

5.8.3. Testing Evidence

# 5.9 The system will display commands on screen during video playback

## 5.9.1. Requirement

The system will display the currently being executed command in the video feed as well as recordings.

## 5.9.2 Testing Process

1. The user will press the toggle video button
2. The user will start a recording
3. The user will issue a command to the system
4. Upon completion of the command, the user will stop the recording
5. The user will check the recording for executed command

## 5.9.3. Testing Evidence

5.9.3.1 Testing Evidence

# 5.10. References and File Links

## 5.10.1. References (IEEE)

[1]StepperOnline,LRS-100-12 MEAN WELL 100W 12VDC 8.5A 115/230VAC Enclosed Switching Power Supply, 2022,Online,[Available]. https://www.omc-stepperonline.com/switching-power-supply/lrs-100-12-mean-well-100w-12vdc-8-5a-115-230vac-enclosed-switching-power-supply-lrs-100-12

### 5.10.2. File Links

## 5.11. Revision Table

| | |
|---|---|
| 05/03/2022 | Dennis: Revised and edited section 5 and section 2. |
| 05/03/2022 | Kevin: Revised testing evidence |
| 05/03/2022 | Ryan: Clarified Testing requirements |
| 05/03/2022 | Felipe: Updated Sections 5.2-5.9 |
| 04/22/2022 | Felipe: Updated system requirements (Section 2 and 5) |
| 03/06/2022 | Felipe: Updated section 5.6 - 5.8 |
| 03/06/2022 | Dennis: Worked on editing and revising section 5, transferred requirements from section 2 |
| 03/05/2022 | Kevin: Wrote brief paragraphs for 5.1 and transferred requirements from section 2. |

# 6. Project Closing

## 6.1. Future Recommendations

### 6.1.1. Technical recommendations

6.1.1.1 The PWM signal for the tool head servo could have some signal integrity issues. To resolve this issue the signal could be routed to a different location than the bottom of the PCB. The signal route is next to the 5 volt route and ends at the 12 volt plane. So the signal route can be moved to be more isolated and to a different position either to the left or right. This will also help wire management since the PCB would not have connectors on all four sides.

6.1.1.2 Our project began in the middle of the Covid19 pandemic. During this pandemic many IC chips and tools needed were out of stock. Because of this the team had very limited choices in microcontrollers and chips. Over time our black Pill microcontroller started to get more inconsistent and in some cases not work. The team has only one of them and if we had extras we would have less hardships. During tests with the 5 Volt circuit the main chip was destroyed. Luckily the team has one more extra, which also died. The 5 Volt regulator in use had internal regulation, so the diode was inside the chip. We do not recommend this chip since its very hard to track the change in voltage

and to debug electrical errors. Both the Black Pill and 5 Volt regulator chips are now out of stock.  When shopping, the team did not buy enough extras and are now paying the price. It is recommended to buy more than needed microcontrollers and IC chips in case something breaks or malfunctions.

6.1.1.3 To allow for more customizability of the User Interface, using Glade is not recommended since it locks features that may want to be used. Instead, it may be better, for the long run, to write the xml by hand or to build the interface during run-time. It may take longer initially, but it is much easier to change later on.

6.1.1.4 To further improve the machines rigidity and robustness, using a material that is more durable would offer a large improvement in the mechanical structure. Due to the fast turnaround times that are required for this project and the limited amount of resources that we are able to access, most of the complex and specialized parts are 3D printed, which are not always the most durable pieces. To improve that, using custom machined materials such as aluminum or steel can fix mechanical problems that could occur on the system.

### 6.1.2. Global impact Recommendations

6.1.2.1 To reduce pollution from e-wastes teams should be encouraged to do multiple designs review for PCBs. Our group went through multiple iterations of PCBs which left us with old PCBs boards that will not be used. Aslo wire runs should be planned out to reduce the amount of wire that is used. The team made mistakes where wire runs were too short or too long. This meant that some wires had to be remanufactured causing more wastes. Another way to reduce wastes is to have the microcontroller and stepper drivers on the PCB itself. Due to shortages this team was not able to accomplish that. As a result, our PCB became a motherboard which increased the size of the PCB.

6.1.2.2 Designing the system to have an extended and serviceable operating life cycle would help to keep as much waste out of landfills as possible. By using recyclable parts, we can also reuse the parts that we make incase of failure or recycle them properly.

### 6.1.3. Teamwork Recommendations

6.1.3.1 When splitting the responsibility for blocks our team split the mechanical structure and enclosure between two different members. This leads to lots of communication problems where a block would change a little and the other block would have to change as well. This would cause more confusion and work. To streamline this process the team should have kept one person to mechanical structure and enclosure. Doing so would have caused less confusion and a more efficiency to the time spent.

6.1.3.2 The project cannot be completed without communicating with teammates and that includes our project partner. Throughout the project the team had a bit of trouble scheduling online team meetings with our project partner, since Textronix could not meet

after 5:00pm. This caused the team to have meetings in the early morning or in between classes. In the future it is best to leave more room open for scheduled meetings to better time management for the whole team.

## 6.2. Project Artifact Summary With Links

### 6.2.1 PCB and Schematic Diagrams

6.2.1.1 PCB Schematic PDFs

### 6.2.2 Mechanical Structure Fusion360 Designs

6.2.2.1 Gantry and Motor designs
6.2.2.2 Bracket 3D Printing Models

### 6.2.3 GRBL HAL file for Black Pill

6.2.3.1 GRBL HAL Code

### 6.2.4 Enclosure Models

6.2.4.1 Enclosure
6.2.4.2 System PCB
6.2.4.3 Raspberry Pi 4

### 6.2.5 User Interface / Hardware Control API Source Code

6.2.5.1 User Interface and HCAPI Code (WIP)

## 6.3. Presentation Materials

6.3.1 Poster

## 6.4. References and File Links

### 6.4.1. References (IEEE)

### 6.4.2. File Links

## 6.5. Revision Table

| | |
|---|---|
| 05/06/2022 | Dennis: Added artifacts and future recommendations |
| 05/06/2022 | Felipe: Updated Section 6.1.1.3 and 6.2.5.1 |
| 05/06/2022 | Kevin: Added artifacts files |
| 05/06/2022 | Ryan: Added formatting, recommendations, and Artifact files |
| 05/05/2022 | Kevin: Wrote one recommendation per subtopic |

# A. Appendix