MAVinator
Design Document

Team: SDMay25-15

Advisor/Client: Dr. Tayeb

Team: Luke Post, James Peterson, Nathan Reff, Daniel Ripley

Email: sdmay25-15@iastate.edu

Website: sdmay25-15.sd.ece.iastate.edu

Executive Summary

The Center for Nondestructive Evaluation (CNDE) lab plays a critical role in examining and evaluating metals and other materials to determine their safety and suitability for research, development, and various applications. By using advanced non-destructive testing techniques, the lab identifies cracks, flaws, or other potential defects in materials before they are used in industrial or governmental applications. This evaluation process is essential in ensuring that materials meet safety and reliability standards.

The lab has few scanning products and has tasked us with building a new scanner that can be used by the professors and members of the lab. The goal of this project is to build a 3D (XYZ) scanning platform for millimeter wave imaging with a similar user interface to Klipper or Octoprint.

The key design requirements are that the scanner must have a motion volume of 300 mm x 300 mm x 300 mm or larger, and a positional accuracy of 0.5 mm. In order to accomplish this we are modifying an open source Voron 3D printer with an in-house designed millimeter wavelength PCB sensor set. As such, the 3D scanner utilizes a stepper motor and belt-driven gantry design. The approach for this part of the project was fairly simple, we followed a build guide for implementing the Voron 3D printer, along with manufacturing the PCB sensor set ourselves in-lab. This greatly reduces the cost of the design and allows us to focus on the physical and user interface.

The graphical user interface has the following features:

- General Movement
- Perform automated scans on a uniform cartesian grid.
- Perform data collection from a millimeter-wave device.
- Process the data using SAR algorithm and display the results.

The core functionality of scanning and data processing has been implemented and tested successfully. The user interface has a clean design and provides all of the necessary functionality to perform a scan and understand the status of the scanner. This was done through the use of html, css, and js for the frontend and using flask (python microframework) for the backend, along with integrating Marlin firmware into the motion controller.

In order to access the user interface the user must simply input the device's IP into their browser with the port 5000 (<deviceIP>:5000). Our client has stated that this qualifies as a Minimum Viable Product, but there is much to be improved upon.

The next steps will be to refine the toolhead system, post-processing options, and user interface further. No additional hardware or software is required for this, simply more time and research are necessary to accomplish these goals. It would also be good to implement mesh bed leveling, though this may require additional sensors.

Other Potential Next Steps:

- Allow users to define their own scan patterns.
- Modular sensor classes to support swapping tool heads.
- Add clipping functionality to SAR generation.
- Add filtering functionality to SAR generation.
- Polish user interface.

Learning Summary

The development of the MAVinator scanner provided valuable learning opportunities, allowing the team to apply engineering skills and principles while gaining hands-on experience in hardware assembly, circuit design, software development, and system integration.

Development Standards & Practices Used

- Hardware development Practices
    - Followed structured assembly guidelines using the build guide
    - Ensuring precision alignment during the gantry assembly to meet accuracy requirements.
- Circuit Design Practices
    - Designing custom PCBs with proper routing, grounding, and power management.
    - Applying best practices in wiring such as labeling, bundling, and securing cables to prevent damage or interference.
    - Practice safe soldering techniques
- Software Development Practices
    - Writing clean, modular, and well-documented Python scripts for the backend and data collection.
    - Testing the motion control firmware using iterative debugging and verification procedures.
    - With multiple threads running concurrently ensure safe memory usage at all times.
    - Camel-case naming convention for the most part, with some snake-case for matlab translations.
- Engineering Standards:
    - IEEE standards for embedded systems and software development.
    - ISO standards for accuracy and repeatability in measurement equipment.
    - IPC standards for PCB design and manufacturing.
    - ANSI standards for safe mechanical assembly practices.

Summary of Requirements

- **Build Requirements:**
  - Develop a 3D (XYZ) scanning platform for millimeter-wave imaging by basing the design on and adapting an open-source 3D printing platform.
  - Achieve an imaging volume of at least 300 mm x 300 mm x 300 mm.
  - Ensure positional accuracy of 0.5 mm.
- **Mechanical and Electronic Assembly**:
  - Utilize a stepper motor and belt-driven gantry design.
  - Assemble the mechanical and electronic components of the scanner.
  - Upload and configure the motion controller firmware.
- **Graphical User Interface (GUI) Features**:
  - Enable homing of the scanner.
  - Allow automated scans on a uniform Cartesian grid or user-defined grid.
  - Perform data collection from a millimeter-wave device.
  - Process data using SAR algorithms and display results (MATLAB scripts provided).

Applicable Courses from Iowa State University Curriculum

- *EE 4140 - Microwave Engineering*
  - *This course introduces students to microwave circuit design and testing which was valuable in the assembly of the PCBs*
- *EE/CPRE 3300 - Integrated Electronics*
  - *Circuit design*
- *SE/Com S 3190 + Com S 3090*
  - *These courses cover software development of frontend and backend for designing a website/app using html, css, and javascript*
- *CPRE 3080*
  - *This course was fundamental to understanding system layers*

New Skills/Knowledge acquired that was not taught in courses

This project has provided the team with the opportunity to learn and develop a variety of skills that go beyond normal coursework, offering valuable hands-on experience in real-world engineering challenges

- **Mechanical Assembly**:
    - Assembling a complex gantry system and aligning axes for precise movement.
    - Understanding and implementing 3D printer-based motion platforms.
    - 3D modeling parts built off of other models to be put to use in real life.
- **Raspberry Pi Deployment:**
    - Picking out libraries that support our given architecture.
    - Headless connection methods.
    - Built upon previous Linux experience.
    - Flashing the Pi OS.
- **Circuit Design**:
    - Designing, fabricating, and testing custom PCBs tailored for scanner electronics.
    - 3 out of 4 of us are computer engineering majors and have not touched any type of circuit design like this before
- **Firmware Integration**:
    - Configuring and debugging firmware for stepper motors and motion control systems.
- **Python Development**:
    - Creating web-based GUIs (Web UI) for scanner operation using Python backend frameworks.
    - Interfacing Python scripts with external sensors and hardware.
- **Measuring using a DAQ:**
    - Understanding how the AD2 reads in values, outputs waveforms, can be triggered to take a measurement, and then applying all of this in python through their SDK.
- **Millimeter-Wave Imaging**:
    - Understanding the principles of millimeter-wave technology and its application in imaging.
- **Project Management**:

- Coordinating multi-disciplinary tasks such as mechanical assembly, electronics integration, and software development.
- Documenting processes and results for academic and professional purposes.
- Group scheduling and developing a consistent schedule to meet amongst ourselves and with our client.

Table of Contents

1. Introduction

1.1. Problem Statement

Scanning can be a time consuming process and there are often not enough scanners to go around. Simply buying a scanner would be an option if they were not exorbitantly expensive. Fortunately with a single **M**illimeter sensor **A**rmed **V**oron(**MAV**inator) we can build a cheap scanner with a large scan volume. With the addition of a simple user interface that can be remotely accessed; Scanning at Center for Non-Destructive Evaluation (CNDE) will become better than it ever has been for the technicians, leadership, and clients.

1.1.1. Project Narrative

Everyone here at the CNDE is well aware of the shortage of millimeter wavelength scanners within our facilities. Lab technicians have to work harder than ever to ensure their scans in a timely manner so that they do not interfere with others using the facilities. This is a problem as millimeter wavelength scanning can safely reveal obstructed and less than visible details of a medium sized object. In addition to being cool, a lack of access to these scanners can result in further project delays due to increased difficulty in troubleshooting and evaluation. More scanners would have been purchased long ago if it were not for the exorbitant cost of a packaged system.

So the CNDE is indeed in need of a cheap and effective millimeter wavelength scanner solution. That is where the **M**illimeter wavelength **A**rmed **V**oron(**MAV**inator) scanner comes in as a viable solution going forward. This system makes use of the open-source Voron 3D printer motion system, an in-house millimeter wavelength scanner PCB, its sister control board, and our design of the physical and digital user interface to allow for a cheap and effective machine with a large scan volume and simple user controls. If the first build proves the concept, then this system could be implemented on a larger scale as well due to the low cost.

Our project is bringing the MAVinator to fruition and doing it well. A lot of the quality of the scan will hinge upon the quality of the printer build, PCB testing, and programming of the system, so we will do our best to document the process and any areas of improvement to further refine our process. If we

are successful we will have a novel fully functioning non-destructive scanner operating within the millimeter wavelength range (119-134GHz) for the lab to make use of.

1.2. Intended Users

1.2.1. Background of Users

Our users have a variety of requirements, but many of them share some common needs. One of these common requirements is that the scanner must operate with millimeter waves. Another common requirement is a reasonably short scan time to promote efficiency. It also must be able to scan a 300mm x 300mm x 300mm region. Additionally the product should look professional and have a user interface that is easy to understand, and overall easy to operate. We have created some different personas that represent the different users and their needs that this project aims to address.

1.2.2. Four Types of Users

1. Lab Technician

Eli needs to be able to scan materials at a quicker rate because he is experiencing too much downtime in his project which is leading to unmotivated work and adding another scanner could do that.

He also wants to be able to extend upon the research he does in the lab to higher frequencies which is why a millimeter wave scanner is important.

Requirements:

- Functional
  - Needs a scanner that works in the millimeter wave frequency range
  - Needs the scanner to be able to move in 3 dimensions
- Resource
  - This system needs to be able to connect to a web app or computer to control
- Physical

    - Should be large enough to scan the things his boss gives him (which will be a max size of 300 mm x 300 mm x 300 mm)
- ➢ Aesthetic
    - The app should look good enough that it is easy to use and understand
- ➢ User experiential
    - Needs the software to be easy to use either from a web app or a computer connected to the device
    - Needs it to export a file of the data to be analyzed

2. Governmental Clients

As an investigator at NASA Magnum needs to reveal the internal structural makeup of his custom manufactured item because he needs to be able to make a more informed decision based on that.

Requirements:

- ➢ Functional
    - Reliable and repeatable results
    - Analysis even through opaque materials
    - Non destructive investigations
    - Reasonable scan times
- ➢ Resource
    - Time it takes to perform the scan is valuable to this type of user
- ➢ Physical
    - Maintain the safety and integrity of the item to be evaluated
    - Could need anywhere from 1cm x 1cm x 1cm to 30cm x 30cm x 30cm or possibly larger
- ➢ Aesthetic
    - High fidelity scan results
- ➢ User experiential
    - Simple ordering experience

3. Private Clients

Ted needs a way to seamlessly integrate reliable 3D millimeter wave scanning hardware and software components into advanced security systems because this ensures precise sensing capabilities and simplifies product development

for WaveSense Innovations, keeping their solutions at the forefront of the industry's technological advancements.

Requirements:

- ➢ Functional
  - It needs to be able to do millimeter wave scanning
  - Scanner must be able to identify various materials within scanning area
  - System should provide API for integration with other applications
- ➢ Resource
  - It should be cheap to build
  - Scanner should not require more than 4 GB of RAM
- ➢ Physical
  - Could need anywhere from 30cm x 30cm x 30cm to 1m x 1m x 1m or possibly more
  - Total weight should not exceed 5 kg for ease of portability and installation
- ➢ Aesthetic
  - Scanner exterior should have modern design
  - App should have a sophisticated design and be user friendly
- ➢ User experiential
  - Company associates should be able to operate easily
  - Software interface should be intuitive with clear visual indicators and real-time feedback
- ➢ Environmental
  - System should be able to operate indoors and outdoor environments

4. Senior lab Technicians

As a lead researcher at CNDE, Tabey needs to look deeper inside small volumes of material for her own research and vicariously through her team for larger projects. Tabey needs a more affordable scanner in the CNDE lab because the number of scanners in the lab is too few to effectively complete work.

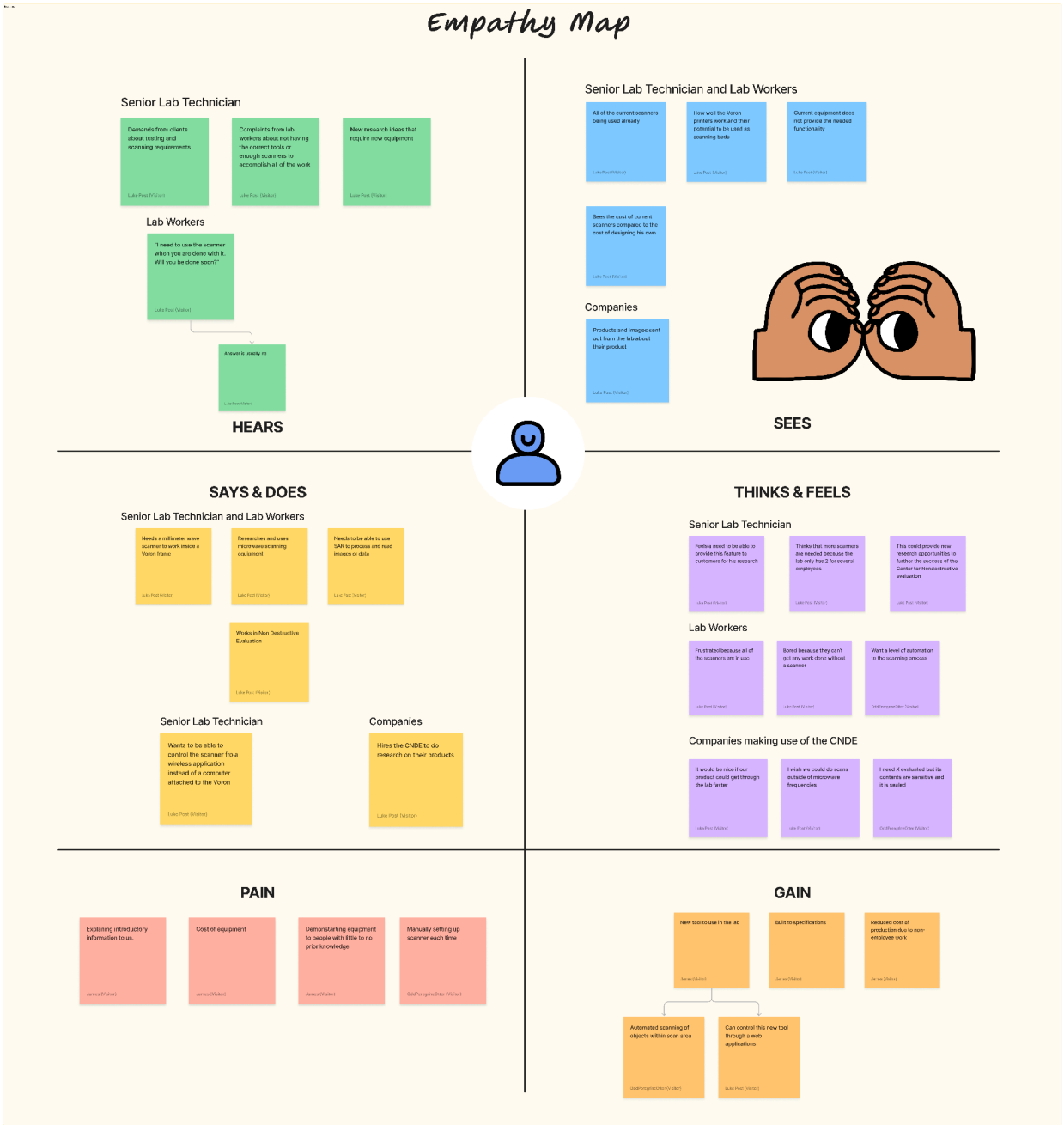
Requirements:

- ➢ Functional

    - It needs to be able to do millimeter wave scanning
    - It needs to fit within the existing lab environment
- Resource
    - It should be cheap to build
    - Build time should not be longer than one month
    - The time it takes to operate should be the same if not less than other scanners
- Physical
    - It needs to be able to be implemented on a Voron printer
    - It needs to cover an area of 300 x 300 x 300 mm
    - It needs to make use of the in house millimeter scanner
- Aesthetic
    - The app should look sleek while still providing good user experience
    - The scanner should look sturdy and professional
- User experiential
    - Lab technicians should be able to operate easily
    - The scanner should be able to be remotely started and stopped

1.2.3. Empathy Map

Empathy Mapping helps identify the thoughts and feelings of a user. As our primary user is a senior Technician, we used him to create our primary empathy map. In this empathy map, we were able to understand how this user will interact with the final product .



2. Requirements, Constraints and Standards

2.1. Requirements and Constraints

2.1.1. Physical Requirements

- The finished product should easily fit into the CNDE lab environment.
- The overall frame dimensions will be 350mm x 350mm x 350mm.
- The design should be compact, stable, and easy to position within the lab setting.

2.1.2. Functional Requirements (specification)

- Scanner should operate within a volume of 300mm x 300mm x 300mm.
- Scanner should be able to detect dense materials at least 2.4mm in width.
- Sensor head should work with the existing toolhead mount and raspberry pi board.

2.1.3. Resource Requirements

- The scanner should be cost effective, utilizing affordable components without sacrificing performance or reliability.

2.1.4. Aesthetic Requirements

- The final product should look professional, clean, like a commercially available scanner.
- The wiring and electronics should be hidden where possible.
- The print head should fit with the aesthetic of the overall build. This is aided by the already professional Voron motion system.

2.1.5. User Experiential Requirements

- The scanner should be designed for ease of use, enabling users to start scans quickly without needing to make physical adjustments.

- Preparation for scans should be intuitive, and the scanning process should be as fast and efficient as possible while maintaining accuracy.

2.1.6. Environmental Requirements

- The design should comply with environmental standards for electronic devices, using materials that are durable yet environmentally friendly where possible.

2.1.7. UI requirements

- The user interface should be intuitive, easy to navigate, and designed to guide the user through the scanning process with ease.
- The UI should present only essential features, keeping the workflow streamlined.
- The design should be visually appealing and cohesive with the professional aesthetic of the hardware

2.2. Engineering Standards

Engineering standards are an essential part of modern design and engineering. Without them the likelihood of two devices using a similar communication protocol would drop drastically. Our team has placed a great deal of emphasis on recognizing and incorporating IEEE standards where possible. Below is an outline of the key standards relevant to our project.

2.2.1. Built-in Standards

Built in standards are standards that are implemented by subcomponents that our team has no hand in designing but will still be pertinent to know.

- 802.11ac (Wi-Fi standard)
    - This standard governs wireless networking and communication protocols, ensuring our scanner integrates effectively with the CNDE lab's existing wireless infrastructure. Compliance with this standard allows for reliable and fast data transmission over Wi-Fi.

2.2.2. Design Standards

Design standards are standards that are not given to us by subcomponents but are selected by design. These are chosen to ensure the scanner complies with standards for devices in a similar class.

- IEEE 149: Standard Test Procedure for Antennas
  - This standard provides test procedures for evaluating antenna performance. This standard is applicable to our project because we will be using an antenna to transmit and receive millimeter waves. Adhering to IEEE 149 ensures that the antenna used in our scanner is accurately tested and optimized for effective signal transmission
- IEEE C95.3: Recommended Practice for Measurements and Computations of Electric, Magnetic, and Electromagnetic Fields with Respect to Human Exposure to Such Fields, 0Hz to 300 GHz
  - This standard addresses the measurement of electric, magnetic, and electromagnetic fields, specifically with regard to human exposure to such fields. This is applicable to our project because we will be using millimeter waves between 119 and 134 GHz
- IEEE 26514: Standard for Adoption of ISO/IEC 26514:2008 Systems and Software Engineering--Requirements for Designers and Developers of User Documentation
  - This standard guides the creation of user documentation for systems and software products. It is applicable to our project for the documentation we create on how to interface with our finished product and maintain it.
- P3397: Standard for Synthetic Aperture Radar (SAR) Image Quality Metrics
  - This standard defines quality metrics for SAR imaging systems. It applies to our project because we will use SAR to process the data and display the results of the scanner

3. Project Plan

3.1. Project Management/Tracking Progress

Project Management Methodology:

We will employ a hybrid approach combining elements of both Waterfall and Agile methodologies to efficiently manage the MAVinator project.

Waterfall Methodology:

- Phase-based: The project will be divided into two distinct phases: Hardware, and Software with the hardware phase scheduled to be completed this semester. The software development phase will consist of developing the user interface and development of the frontend and backend. This portion is anticipated to approach completion towards the end of next semester.
- Sequential: Certain sprints will rely on others being completed before they can begin. The assembly of the sensor tool-head relies on the completion of the housing and PCB. The electronic wiring requires the frame and motors to be mounted to have anything to wire. Lastly the user interface will require all other components be assembled before it can begin.
- Documentation-heavy: Documentation will be maintained throughout the project, including requirements/specifications, this design document, test plans, and user manuals.

Agile Methodology:

- Hardware task decomposition: We have broken down the phases of this project into simpler segments or sprints. The housing for the scanner, the printer frame, and the PCB as smaller components of the Hardware phase can be worked on in parallel.
- Software task decomposition: Similarly we broke down the software development into smaller sprints. The HTTP development of a connection between frontend and backend, the BTT Pi imaging, websocket integration, frontend polishing, and DAQ + SAR algorithms were our major sprints.

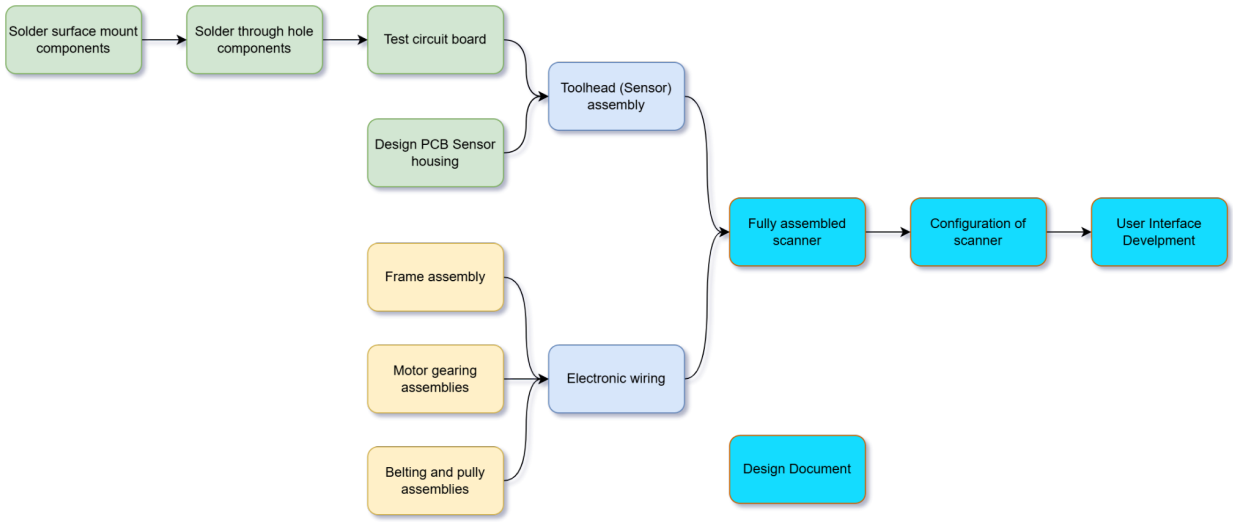
- Frequent feedback: Regular feedback loops have been established with the project stakeholders to ensure that the project is aligned with their needs and expectations.
- Continuous improvement: We have continuously evaluated and improved our processes and methodologies throughout the project. This has been accomplished through team meetings, and independent research.

We primarily utilized discord for communication and tracking progress, such as notes from meetings, progress pictures, and any additional documentation/research. We also divided up work and decided on the leaders for each milestone over Discord calls together.

We primarily used github for the software development portion of our project. Uploading firmware, frontend code, and flask python backend code, with any other additional files.

3.2. Task Decomposition

3.2.1. Hardware Task Decomposition

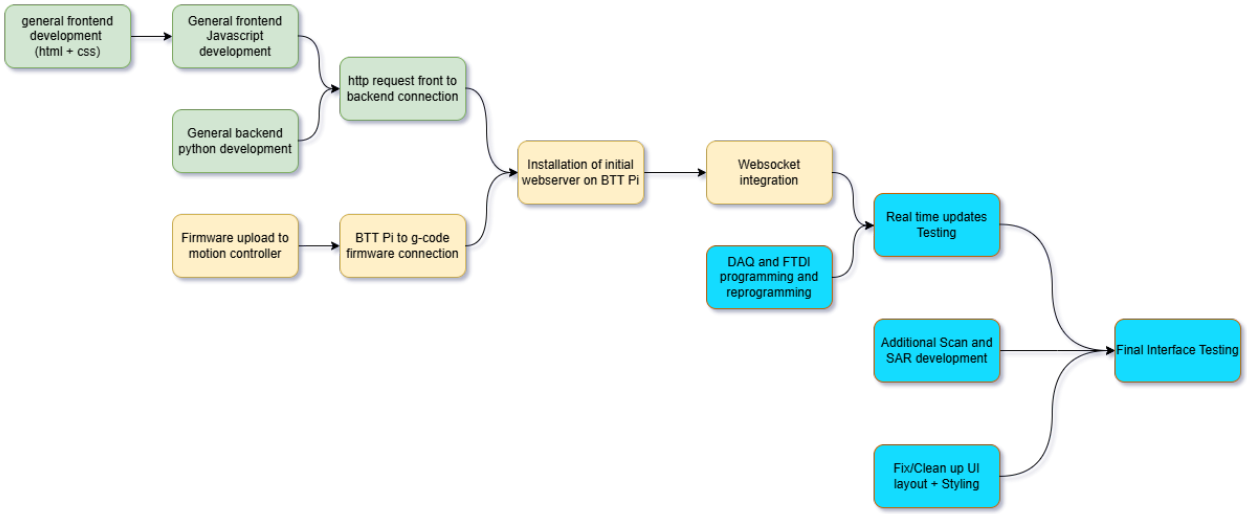


Semester one consisted of the hardware integration. The task decomposition starts with the foundational PCB, soldering all components, followed by testing the circuit board. The PCB is then ready for the design and print of a housing/mount, after which the sensor toolhead is ready for assembly. Concurrent to those, the frame and drives are assembled, once they're ready,

the wiring will be integrated. With those two major milestones reached the MAVinator is ready for full assembly and our goal for this semester is complete. The design and implementation of a software user interface can then begin. All throughout the design documentation is being updated, referenced, and polished.

3.2.2. Software Task Decomposition

This is the general layout of the software development process of the project including the connection from the backend to the frontend through websockets and HTTP requests



The task decomposition of the software development process starts with first making the initial html, and css files for a general user interface. The next big step is the client to server connection using javascript for the frontend and the using flask for the backend. While this is happening, uploading the firmware to the Octopus main board and testing sending g-code to the scanner for general movement and homing functionalities is ongoing. Once firmware is on the Octopus an image can be installed on the BTT Pi. After all of these steps are completed installing the backend software onto the BTT Pi allows for a connection from the user interface to the scanner, thus sending g-codes from pushes of ‘buttons’ on the web GUI. Lastly finalizing the development of Scanning and SAR algorithms along with cleaning up the UI to be User friendly, and testing everything along the way the foundation of the software side of the project is complete.

3.3. Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestones 1 encompasses all work needing to be completed before semester two, and Milestones 2 encompasses all following work.

3.3.1. Milestones 1:

1. PCB Soldered and ready for testing
2. Scanner frame assembled with less than 3mm out of square
3. Scanner hardware interface ready for PCB mounting
4. Testing of PCB radar completed
5. Wiring of motion system completed
6. Sensor mounted, wired, and working on laptop backend
7. Systems testing started

3.3.2. Milestones 2

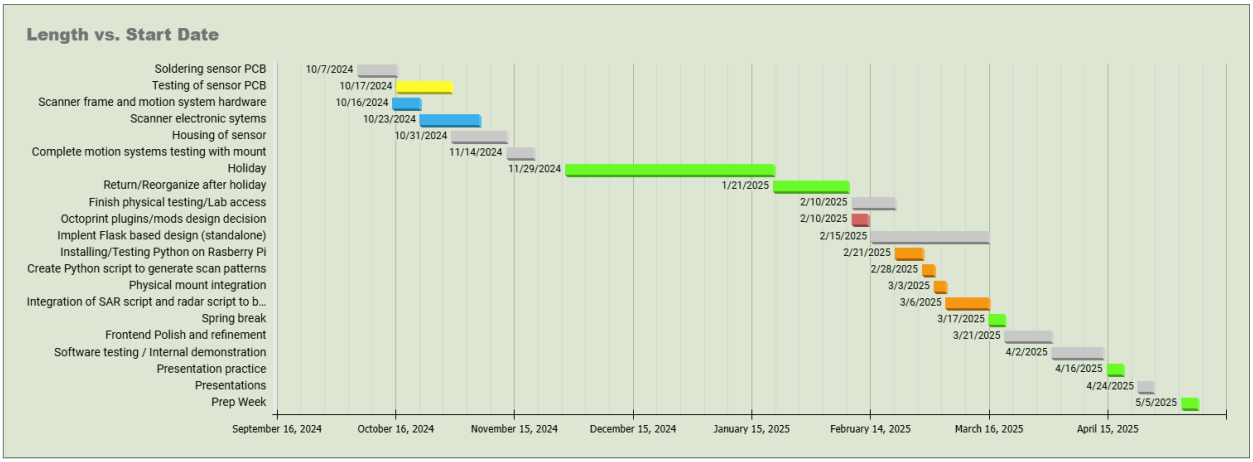
1. Firmware installed onto Octopus Motion board and basic system testing completed
2. Finalization of software design architecture decisions and basic Flask web UI implementation
3. First version of web UI installation and testing on BTT Pi hardware
4. Scan pattern generation integrated into scan function and outputs an array of g-code commands to specification
5. Physical mount redesign to accommodate radar swap fits new radar and gantry shuttle
6. Radar scripts re-written to read single shot data read from new DAQ and ensure FTDI scripts write PLL registers on the BTT Pi
7. SAR algorithm processes .scan files similarly to matlab and file-system scripts save .scan files where the user selects
8. Final testing + Presentation, the general sharing of insights gained and success achieved

3.4. Project Timeline/Schedule

This is the schedule the project has followed for the hardware development portion. The scanner electric wiring end date has had to be pushed out by one week, resulting in an overlap with the housing design.

Milestone	Start Date	End Date	Length	Progress/status	Lead	Notes
Soldering sensor PCB	10/7/2024	10/17/2024	10	100%	Luke	The spi connectors were originally put on reversed
Testing of sensor PCB	10/17/2024	10/31/2024	14	100%	Luke, James	
Scanner frame and motion system hardware	10/16/2024	10/23/2024	7	100%	Nate, Daniel	Two of the rails, one z one y are short bearings
Scanner electronic systems	10/23/2024	11/7/2024	15	100%	Nate, Daniel	
Housing of sensor	10/31/2024	11/14/2024	14	100%	Daniel	
Complete motion systems testing with mount	11/14/2024	11/21/2024	7	100%	Collective	Thanksgiving break after
Holiday	11/29/2024	1/21/2025	53	100%	Collective	Holiday
Return/Reorganize after holiday	1/21/2025	2/9/2025	19	100%	Collective	Back from Holiday
Finish physical testing/Lab access	2/10/2025	2/21/2025	11	100%	Nate + Dan	
Octoprint plugins/mods design decision	2/10/2025	2/14/2025	4	100%		We will develop a webserver in python
Initial design of User interface - Frontend	2/10/2025	2/14/2025	4	100%	Nate	Html + css + js design
Implent Flask based design (standalone)	2/15/2025	3/17/2025	30	100%	James + Luke	Specifically the UI elements necessary for project requirements
Installing/Testing Python on Raspberry Pi	2/21/2025	2/28/2025	7	100%	Dan + Nate	
Create Python script to generate scan patterns	2/28/2025	3/3/2025	3	100%	Luke + James	Fired after each movement to a new point

Physical mount integration	3/3/2025	3/6/2025	3	100%	Daniel	
Integration of SAR script and radar script to backend	3/6/2025	3/17/2025	11	100%	Luke + James	Pre+Post-scan processing
Spring break	3/17/2025	3/21/2025	4	100%		
Frontend Polish and refinement	3/21/2025	4/2/2025	12	80%	Nate + James	
Software testing / Internal demonstration	4/2/2025	4/15/2025	13	0%	Daniel	
Design Documentation	4/16/2025	4/20/2025	4	0%	Collective	
Poster Design + Practice	4/24/2025	4/28/2025	4	0%	Collective	
Prep Week + Presentations	5/5/2025	5/9/2025	4	0%		



3.5. Risks and Risk Management/Mitigation

3.5.1. Key Risks:

The "Key Risks" section is crucial for identifying and mitigating potential issues that could hinder the project's success. By proactively recognizing these risks, the team can develop strategies to minimize their impact and ensure the MAVinator project stays on track. The risk matrix is a more graphical way of visualizing their categorization.

Risks - Core Details

Rank & Trend	Risk Title	Approach	Likelihood	Consequences
1	PCB does not work as intended initially	M	5	3
2	Physical build runs over schedule	W	2	4
3	Linear rail missing bearings	A	5	1
4	PCB has catastrophic short	A	1	5
5	Frame out of square during testing	W	2	1
6	Sensor toolhead cannot determine location	M	1	5
7	Over-voltage motors	R	1	2
8	Electrical interference	M	3	4
9	Data could be lost during a scan	M	3	3
10	User entered Z-height too low	M	3	5

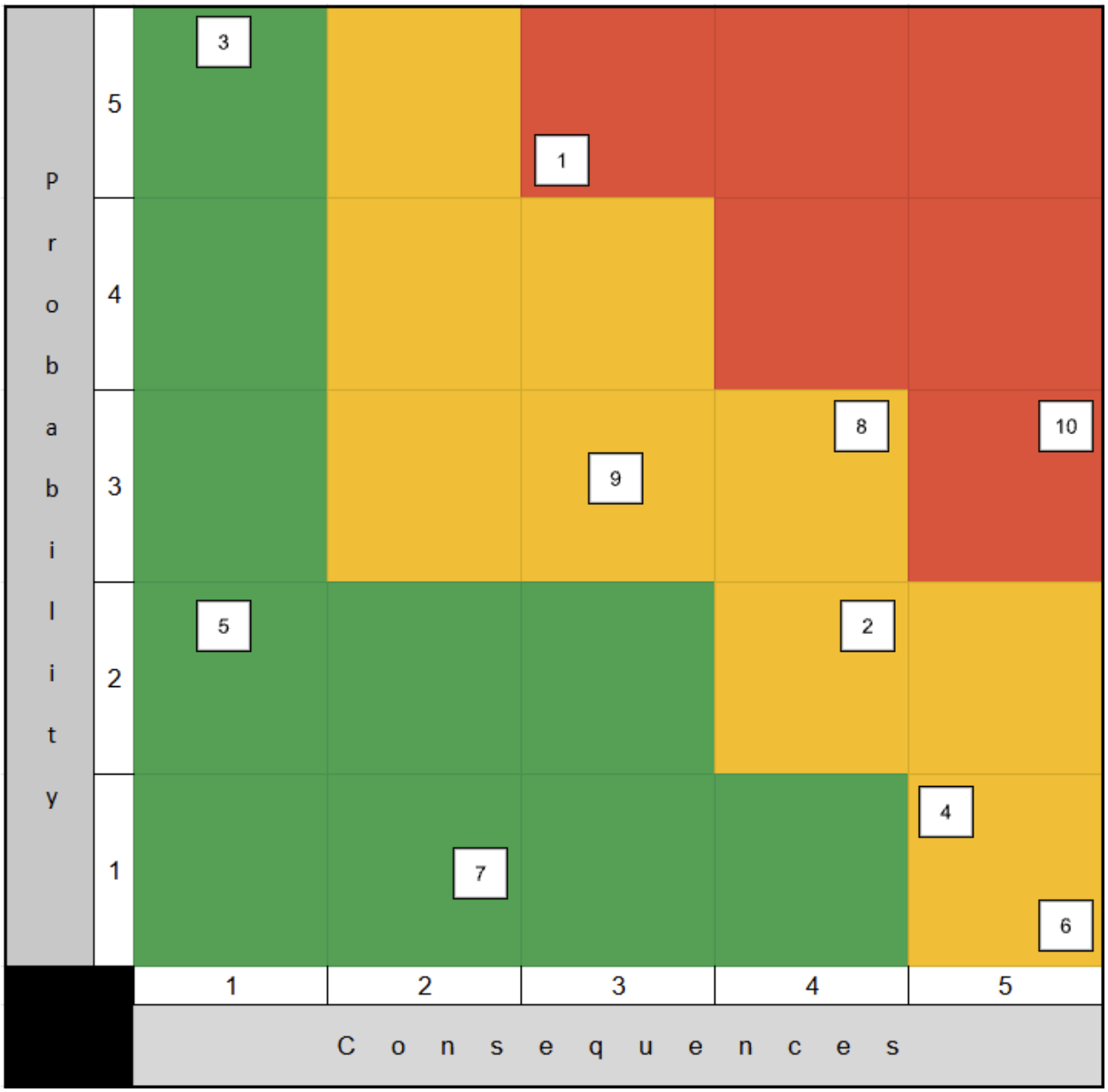
Risk Assessments - Descriptions & Mitigations

Rank & Trend	Risk Title	Description	Mitigation Strategy

1	PCB does not work as intended initially	A fixable issue occurs with PCB sensor, could not scan	
2	Physical build runs over schedule	Build runs into next semester, could not scan	
3	Linear rail missing bearings	One linear z and y rail are missing <4 bearings, accepted, plan to replace	
4	PCB has catastrophic short	A short bad enough it burns the board irreprably, could not scan	
5	Frame out of square during testing	Frame out of square/gantry not de-racked results in inaccuracies, could scan	
6	Sensor toolhead cannot determine location	Sensor loses or has no way to determine position, could break sensor PCB	Emergency stop on all pages
7	Over-voltage motors	Motors run over-voltage due to improper driver config, burnt out, could scan	
8	Electrical interference	Interference impedes accurate measurement, need to eliminate internal sources	Electrical shielding on all A/B motor wires

9	Data could be lost during a scan	Data lost on page refresh (front-end) or power loss (back-end)	Save data to file as read, process on backend
10	User entered Z-height too low	Low Z-height could damage object or sensor	Enforce minimum Z-height at all times

3.5.2. Risk Management Matrix



Risk Assesment

# Rank & Trend	Risk Title	Approach	# Likelihood	# Consequences	Description	Mitigation Strategy
1	PCB does not work as intended initially	M	5	3	A fixable issue occurs with PCB sensor, could not scan	
2	Physical build runs catastrophically over schedule	W	2	4	Build runs into next semester, could not scan	
3	Linear rail missing bearings	A	5	1	One linear z and y rail are missing <4 bearings, we have accepted this and plan to replace, can scan	
4	PCB has catastrophic short	A	1	5	A short bad enough it burns the board irreprably, could not scan	
5	Frame out of square durring testing	W	2	1	Frame being out of square and gantry not being properly de-racked results in inacurracies, could scan	
6	Sensor toolhead can not determine its location	M	1	5	Sensor looses or has no way to determine its position, would probably break the sensor pcb	Emergency stop on all pages
7	Over-voltage motors	R	1	2	Motors are ran over-voltage due to improper Stepper Driver configuration, burnt out, could still scan	
8	Electrical interferance	M	3	4	Elecitrical interferance can impede an accurate measurement, we need to elminate as much internal interferance as possible	Electrical sheilding on all A/B motor wires
9	Data could be lost durring a scan	M	3	3	If data processing is done on front end it is lost on page refresh. If on backend it will be lost on power loss	Save data to file as it is read, process on backend
10	User entered Z-height could be too low for sensor	M	3	5	If Z-height is too low sensor could damage the object to be scanned or the sensor itself	Enforce minimum Z-height at all times
11						
12						

Approaches:
M - Mitigate
W - Watch
A - Accept
R - Research

3.6. Personnel Effort Requirements

3.6.1 Hardware

Milestone	Person-hours
Soldering sensor PCB	15
Testing of sensor PCB	10
Scanner frame and motion system hardware	24
Scanner electronic systems	16
Housing of sensor	12
Complete systems testing	8

3.6.1 Software

Milestone	Person-hours
Frontend UI development	15
Main backend flask development	55
New mount customization and wiring setup	10
Connection to Raspberry Pi	10
Scan pattern generation	10
Firmware + backend research	13
Firmware customization	10

FTDI from BTT Pi troubleshooting	10
Digilent DAQ programming + Troubleshooting	20
SAR Script Migration	35

3.7. Other Resource Requirements

3.7.1 Hardware

Part	Quantity
Millimeter wavelength transceiver board	1
Transceiver control circuit board	1
Voron printer kit	1
Raspberry Pi	1
Computer	1
FTDI cable	1
Coaxial cable	4
Loctite	1
3D printer (to manufacture mount)	1
Digilent Analog Discovery 2 + BNC Breakout board	1

Part	Quantity
Cable Management Kit	1
USB A (male) -> USB C (male) cable	1
Faraday insulation	10ft

3.7.2 Software

Python Libraries Used:

Library	Use
flask	Build web applications and APIs in Python
semantic	Design user interfaces for web applications (HTML/CSS/JS framework)
pydwf	Control Digilent WaveForms hardware devices (like Analog Discovery) from Python
serial	Communicate with devices over serial ports (COM, tty).
ftd2xx	Interface directly with devices using FTDI USB chips via D2XX drivers
matplotlib	Create static, animated, and interactive plots and visualizations
numpy	Perform efficient numerical computations using arrays and matrices
Marlin (firmware)	Uploaded to motion controller for proper sensor movement

4. Design

4.1. Design Context

4.1.1. Broader Context

In a broader context, the MAVinator is designed for any NDE community that is seeking to utilize millimeter wave imaging. Not only does this affect the specific NDE industry, but also all of the industries that rely on NDE to ensure product safety such as the automotive industry, space industry, Navy, and many more. Equipment such as the MAVinator is utterly important due to the enhanced safety it brings to those industries. Faults and defects are able to be detected prior to the product entering the market while saving the companies money as this method of examination is noninvasive and leaves the product perfectly functional.

Area	Description	Examples
Public health, safety, and welfare	The main purpose of the MAVinator is to detect faults, cracks, and any other defects in products before they reach the market or are used for their intended purpose. Being able to discover these things prior to use ensures damaged products don't leave the manufacturing line and provides more safety in every industry that it is utilized.	A Chinese commercial rocket had a defect in the foam insulation which fell off on launch and caused rocket failure (i.e. blow up). This could have been detected using millimeter wave scanning devices.
Global, cultural, and social	The MAVinator is designed around meeting the goals of the industry, which are to provide reliable methods of scanning objects to ensure their safety and functionality.	Development and operation of the MAVinator will allow NDE researchers and scientists the ability to perform scans of materials that returns accurate results while minimizing opportunities for human error.

Environmental	The MAVinator can have a great impact on the environment as it tests objects without destroying them and is predominantly made from 3D printed parts.	Decreases the amount of waste from safety testing in the automotive and space industry as all of the tested material is still usable.
Economic	Our product, the MAVinator, will speed up scan times and therefore reduce the cost to a consumer. It is also relatively cheap to produce in and of itself which enables more people or companies to utilize it.	The MAVinator is more affordable than other current millimeter wave scanning devices and is open source so other users can build off of it, reducing cost.

4.1.2. Prior Work/Solutions

Millimeter and microwave sensing have been used extensively over the past two decades for military, security, and general radar purposes to detect things invisible to the naked eye. This is due to the wavelengths being able to pass through most porous material, such as fabric, fog, or foam, and show any more reflective, usually metal, object on the other side. The first papers presented in this realm were in 1997 about smart technical guidance systems, surveillance, and concealed weapon detection [2]. Shortly after, the origins of what we now see used for security in airports, the L3 Provision, was presented in 2000. That device uses an array of millimeter wave transceivers operating in the range of 16 - 30 GHz to scan a person in a cylindrical manner. It then performed synthetic aperture holographic methods to reconstruct a high-resolution image of the scan target.

This technology has since been shrunk down to a handheld application called MilliCam by Saadat and Ramanathan, et al. [3] This application uses the synthetic aperture radar (SAR) imaging algorithm to map the reflection to the spatial domain. One of the main challenges in transforming these millimeter sensors into handheld devices that operate in the nearfield is the loss in image quality due to aperture motion. The Fourier transform equation used to do the SAR calculation relies on precisely known coordinates of the aperture location, and any error over a half-wavelength (which is 2.5 mm at 60 GHz) can distort the image. [3] fixes this problem by employing a

co-located optical camera. The Millicam can use this to compute the position and trajectory of the device during a hand-swipe measurement. Squint correction was also used to further enhance the image quality.

For our purposes, we are interested in millimeter wavelength technology specifically for nondestructive Evaluation (NDE). In this field, millimeter wave imaging has been used to detect surface cracks in metal [4], building and concrete infrastructure [5, 6], and composite material assessment [7]. This technology has been advanced by Yalcinkaya, Aydin, and Kara [8] to become simpler and more affordable. In their study, parameterized control over the sampling intervals, scanning aperture, and chirp settings was integrated to eliminate the need for complex processing while still maintaining high-quality imaging. One Tx and one Rx antenna operating in the range of 77 and 81 GHz were mounted on a scanning system that could move both vertically and horizontally. The antennas were then moved in a raster manner to scan the target. SAR processing was then completed on the data. It is important to ensure the spatial sampling interval satisfies the Nyquist sampling criterion to avoid aliasing so ghost targets do not appear. This system also used a graphical user interface (GUI) on a host computer to control the movement and scan settings of the device, as well as to display the SAR results.

Furthermore, the CNDE already utilizes 2D scanning systems using microwave wavelength technology for NDE purposes. These systems use Labview to control the movement of the target while the sensor remains in a fixed location and to collect the data. MATLAB is then used to perform SAR processing on the data resulting in a 2D and 3D image. The CNDE commonly uses these imaging systems to detect surface cracks in metal and defects or abnormalities within foam or other low dielectric materials. Image quality of these scans can be improved by pre-processing the data before SAR calculations are performed. These methods have been shown to easily detect rubber pellets inside of foam, short thin wires, and cracks and defects in a wide range of materials.

4.1.3. Technical Complexity

Our project, the MAVinator, contained many components of technical complexity that challenged every member of the group and helped us to grow in our understanding of engineering. Our work crossed several disciplines, not all of which we were familiar with previously, including mechanical system building, precision electronics assembly and testing, web application creation, including front and back end, device control, data collection and processing, and testing.

The MAVinator began with the build of the Voron printer which consisted of several components and complex mechanics. One of the largest components of the build was the gantry system which controls the position of the sensor. This system involved multiple components on its own which all had to be perfectly and precisely aligned to function correctly. Since this controls the position of the sensor, if anything was out of alignment, it could cause serious errors such as crashing into the boundaries, reporting back incorrect locations, and recording data incorrectly. During the build, changes to certain aspects of it had to be made to align with our project goals. This required much adaptability and problem solving as we figured out what worked and what didn't work in our system. This Voron build also included the integration of electronics, a Raspberry Pi and Octopus MCU, which needed to be able to communicate between the user's input and the function of the Voron.

Digging into the electrical engineering side of things, a pair of PCBs needed to be soldered for the sensor portion of the project. This required precision soldering under a microscope and extensive testing to ensure that each PCB was functioning as expected. This is a critical part of any PCB design in all industries as a single short in an electronic can permanently damage an entire system.

We were also tasked within the scope of the project to create a modern web application used to control the Voron and sensor. This required extensive research into GUI design and aspects of frontend and backend coding. Communication in this part of the project was even more vital than before as none of the members had much or any experience creating a GUI to interact with a physical system which meant we were continually learning along the way. The GUI was designed to meet professional standards in terms of physical appearance, user interaction, and code readability.

Adaptability ended up being a major slogan for our project by the end, especially relating to the device control and data collection. Our project required us to interact with a Raspberry Pi, Octopus MCU, sensor PCB, and a digital acquisition (DAQ) device. All of these needed to be able to communicate with each other and work together to make our project function. This system was up and running, but with two weeks before the project deadline, our sensor was needed for a different project, as well as the DAQ we were using. We were unable to be provided with the same sensor, but a new one was given to us, even though our advisor specified we would be using the other sensor we had built. The new sensor had to be programmed slightly differently, and the new DAQ had to be completely reprogrammed. With only two weeks left, this required quick thinking, determination, and adaptability to get our project up and running.

In the end, all of these components had to be seamlessly integrated together to provide the user with a pleasant experience while also upholding the integrity of the functionality of the system.

4.2. Design Exploration

4.2.1. Design Decisions

The MAVinator project aimed to develop a cost-effective, 3D scanning platform for millimeter-wave imaging. Our initial design decisions were based on specific hardware components and their pre-made software frameworks, carefully selected to meet project requirements and client needs. However, as is often the case in complex engineering projects, we encountered unforeseen challenges that required us to adapt and modify our approach. Notably, late-stage changes in available hardware, specifically the Digital Acquisition (DAQ) device and the radar sensor, necessitated significant adjustments to our design and implementation.

Initially, we needed to design a housing for the sensor PCB. This is important for a couple of reasons. To protect the PCB from the elements to ensure the product works reliably, provide a more professional looking design, and ensure optimal sensor positioning/angling. For this mount, we have significantly deviated from the prototype first designed by Aaron McCarville. This was the optimal solution for us as we could extract all the measurements

directly from the existing STL (file format used to represent 3D models) and start over with them.

The next major component we had to design was a UI to control the scanner. This is needed in order to complete scans and view the results in a human understandable format. For this, we chose to utilize Python as the base language for the backend with libraries to supplement our needs. We chose Python due to its widespread support, clients request, and ease of use, allowing the users to update the UI down the road as well. Flask has been researched as the primary library for the GUI design due to its ease of use and broad support as well. We designed the frontend with html, css, and javascript and set up POST and GET endpoints for frontend-backend integration.

We designed a calibration/testing method for the scanner and sensor. This is vital in completing scans as it will position the scanner in a known position to accurately image an object. We used three limit switches for homing the X, Y, & Z axes. When the home button is pressed in the GUI, the sensor will move all the way in one direction on the x-axis and do the same for the y-axis to get its location in those planes. Then, it will move up until it hits another limit switch at the top of the Voron frame, allowing the sensor to know its relative location at all times.

For scan pattern generation, the decision was made to implement a system that mainly operates within a cartesian grid but could be easily customized. This approach ensures basic functionality while leaving room for expansion. The initial focus is on a grid pattern as it meets the core requirement of systematic scanning. This involves calculating a series of XYZ coordinates within the defined scanning volume, which are then translated into G-code commands for the scanner's motion controller. If the users scan dimensions are not easily divisible by the step size, then we chose to let the user decide how to handle the resolution themselves with a notification and suggestion.

Regarding SAR processing, rather than developing the algorithms from scratch, we decided to port existing MATLAB scripts to Python. This decision was primarily driven by efficiency and accuracy. The MATLAB scripts had already been validated and tested, ensuring the reliability of the SAR processing. To facilitate this, AI tools were employed to assist with the translation process. This approach may have moderately increased the time it

took when compared to rewriting the complex algorithms manually. Utilizing AI for this task seemed to complicate things in some regards, but did help to understand some available tools.

At the last minute we discovered that we needed to switch to a new Digital Acquisition (DAQ) device as the National Instruments (NI) DAQ did not have a library that supported Linux. The NI DAQ originally had pre-written code that would work for our purposes and we had to recreate the functionality for the new Digilent Analog Discovery 2, which now serves as our DAQ. We briefly considered using the AD2 to take the place of the FTDI cable but decided against this in the end due to time constraints.

4.2.2. Ideation

For the UI, we went through multiple iterations of design concepts. One idea was to build off the previous UI using Labview, but this UI was not user-friendly and needed improvements. Our next option was to build the UI from scratch, given our ideas; however, this left some unknown variables for the client and unfamiliarity with the current UI. We then thought that we could build a modification for an existing library like Octoprint or Klipper, though this would impose too many requirements and dependencies on our project. Ultimately we wound up deciding on building a Web based user interface similar to the likes of Klipper using Flask. It would be built in such a way that mimics the physical interfaces of existing lab equipment.

With the scanner body (Voron motion system), we had fewer ideas to work off of as a bulk of the design was set by the kit designer. The bulk of the decisions were between modifications to the kit and the firmware to use on the Octopus MCU. The modifications considered were switching out limit switches and adding cable insulation to motor wiring, and location/firmware definition for Z-max movement limit switch.

At first, we were only looking at the build as laid out by the kit. In the later stages of the scanner's construction, modifications to the kit began to be considered as options. We found the options available to us through discussion with print enthusiasts and research online with the exception of the cable insulation. Ultimately, we decided to get the scanner up and running as simply as possible with a first prototype motion system. The decision to insulate the motor cabling came from an examination of the

pre-existing scanners and a discussion with our advisor. These talks ultimately led us to believe the shielding for the A/B motors would be necessary to mitigate the electric interface in the scan.

The process of deciding a firmware has been a bit of a winding road. Initially, we selected Klipper for the firmware to be installed on the Raspberry Pi, which would then flash firmware to the Octopus MCU. We moved away from this idea primarily because of the existence of Marlin firmware that was used in the previous version of this scanner that could be modified for our purposes. This also shifted our goal of software design as now we would be interfacing with the Octopus running the firmware with the Raspberry Pi acting as a server/controller.

We went through several ideas of how the file management and SAR processing would be handled. Initially we were working towards having the file creation and saving along with SAR processing done on the frontend. This was to minimize the load on the BTT Pi, however, we quickly realized that the frontend would lose the data if the page was refreshed which would not work. In the end we decided on the backend creating and storing one scan's worth of data in a temporary file and the client downloading it. Likewise we elected to handle the SAR processing on the backend, uploading the unprocessed .scan file and generating a SAR image that is transmitted back.

When generating the initial pattern for scanning we took in a lot of user feedback to ensure it would provide the needed functionality. At first we simply rounded the differences if the length or width was not divisible by the step size, additionally, we only had one step size for both length and width. After taking in feedback we implemented a prompt for the user to select what they would like to change the scan parameters to evenly divide, the length or step size for the length. Lastly, we implemented independent step sizes for lengths and widths.

Deciding on an operating system was simple after our research indicated that the lightest weight operating system was a customized lightweight Debian provided by Big Tree Tech (BTT) themselves.

When we learned that the National Instruments (NI) Digital Acquisition (DAQ) device would work, deciding on a new DAQ to use was made easier by Dr. Tayeb’s suggestion of the Digilent Analogue Discovery 2. Though we did need

to do additional research to ensure that it would work on Linux and could serve our purposes.

4.2.3. Decision-Making and Trade-off Tables

Using Previous UI	
Pros	**Cons**
Easy to implement	Not user friendly
Familiar to client	Clunky
Know it works	Not portable
Made in-house	Uses LabView
Building from Scratch	
Pros	**Cons**
Made to our needs	Hard to implement
Customizable	Not familiar
Can work with tools we know	Time-consuming
Can be made with Python	
Can be made easier to maintain	

Utilizing Marlin Firmware	
Pros	**Cons**
Basic version provided to us	Every change requires compilation and flashing
Has better support community	Does not include web interface
Runs only on the Octopus MCU, reducing demand on Raspberry Pi	
Simpler in concept as we can make direct modifications to the firmware	
Utilizing Klipper Firmware	
Pros	**Cons**
Easy and quick updates and/or modifications	Documentation may be out of date
More actively supported	Harder to modify for large changes
Has a web interface	Changes are made via changes to a configuration file
	The nature of Klipper means that it must support needed commands, or require a custom plugin library for support of unusual commands like triggering the sensor.

After weighing the Pros and Cons we decided to use the Marlin firmware due to our advisor/client already having it ready for us to use. We still needed to make modifications to support our style of movement and homing.

Frontend SAR and File system	
Pros	**Cons**
Not hardware constrained	Catastrophic data loss on page refresh
	Browser performance could negatively impact output

Using Debian over Armbian	
Pros	**Cons**
Light weight	No desktop environment
Support direct from manufacturer	

Switching to new Digilent DAQ	
Pros	**Cons**
Works on linux	Rewrite entire reading code
Has a python library that works on linux	Lose the differential input, increase in noise
MAVinator would not require an external computer running windows	

4.3. Final Design

4.3.1. Overview

Our project is to build a 3D scanning platform designed specifically for millimeter-wave imaging, allowing users to capture detailed internal aspects of an object. Similar in principle to a 3D printer, this machine will be a scanner that will move in three dimensions: X, Y, and Z directions. Instead of printing, our scanner will use a specialized millimeter-wave sensor device to capture internal aspects of objects in its imaging area.

The frame of the scanner allows for 300 mm^3 of space across all directions. The movement of the scanner head will be powered by stepper motors and belts, to allow for a smooth and accurate scanning movement. The motors are small motors that rotate with the belts connected to them to allow for the movement of the gantry system.

The central portion of the machine is the millimeter-wave imaging device, which will collect data by sending out and receiving millimeter-wave signals. The data from this device lets us create a 3D map of the object. This wound up needing to be replaced with a new radar as it was required for other purposes in the CNDE.

In addition to the physical portion of this project, a GUI is developed to allow users to interface directly with the scanner on a web based application. The users have the ability to move the sensor in any direction, go to certain coordinates, or home any axis or all the axis

The other main functionalities are the ability to scan an object by applying scan configuration values and also viewing real time data. Lastly they are able to upload a .scan file as scan data and calculate SAR.

4.3.2. Detailed Design and Visual(s)

The detailed design is separated into two different sections: Hardware and Software. The Hardware section covers the physical build of the scanner and the integration of the gantry system and electronics. The Software section covers all the files created and the creation of the user interface and the connection from the application to the scanner along with the functionalities.

1. Hardware

Frame

Our design plan requires us to start with the assembly of the frame for the scanner. Extra Care is taken to ensure that the corners are Square from the start of the build. We define our X, Y & Z. This is shown in **Figure 4.3.2-1.**

Motors

The z-motors are placed under each of the bottom corners, working as legs for it (**Figure 4.3.2-4)**. These motors are connected to a drive train gear reduction, the output of which is connected to the corners of the gantry. This allows the gantry to move along the Z axis as the belts move through the idlers.

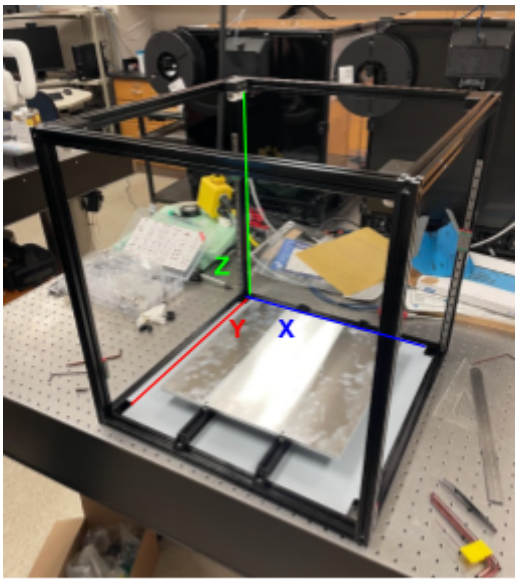


Figure 4.3.2-1: Scanner Framing

Gantry System

The X and Y rails are implemented on the inner workings of the printer, creating the gantry system. A/B belts run along each rail, which allows the sensor head to move in the X and Y-axis while the entire gantry system moves vertically. Both A/B are connected directly to a gear on the motors, then attached to the idlers and toolhead mount to be able to pull it one direction or the other. This happens with a differential between the motors.

When there is a differential between them the toolhead moves in the Y-axis and if they are moving simultaneously then the toolhead moves in the X-axis (see Reference Mechanism: **Figure 4.3.2-3**).

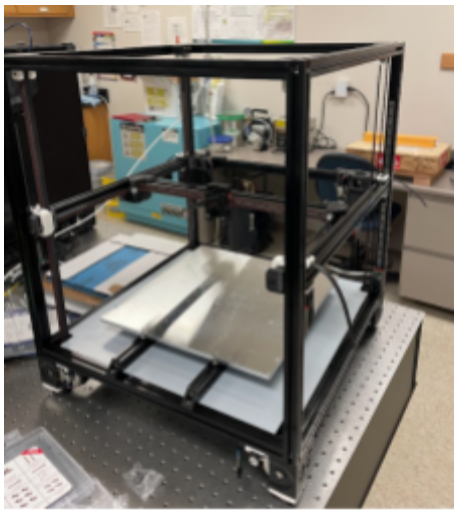
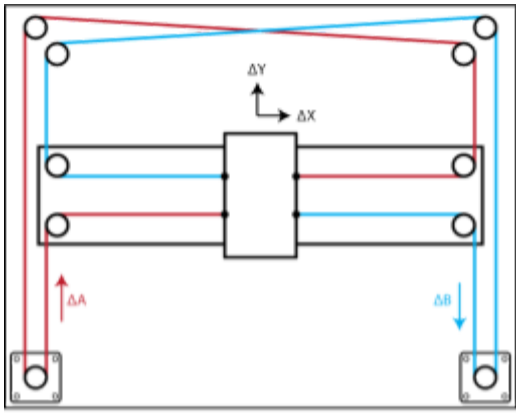


Figure 4.3.2-2: Angled View



Equations of Motion:

$\Delta X = {}^1/_2(\Delta A + \Delta B), \quad \Delta Y = {}^1/_2(\Delta A - \Delta B)$

$\Delta A = \Delta X + \Delta Y, \quad \Delta B = \Delta X - \Delta Y$

Moyer, Ilan E. "Reference Mechanism." *CoreXY*, 2012, corexy.com/theory.html.

Figure 4.3.2-3: Equation of Motion

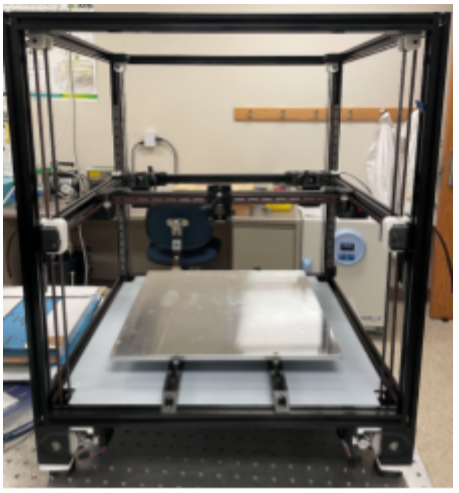


Figure 4.3.2-4: Front Face View

Electronics

On the bottom of the scanner is where all the electrical parts are located. These parts include a Raspberry Pi, controller board, 24V Power Supply Unit, and Power inlet. These will collectively control the Scanner.

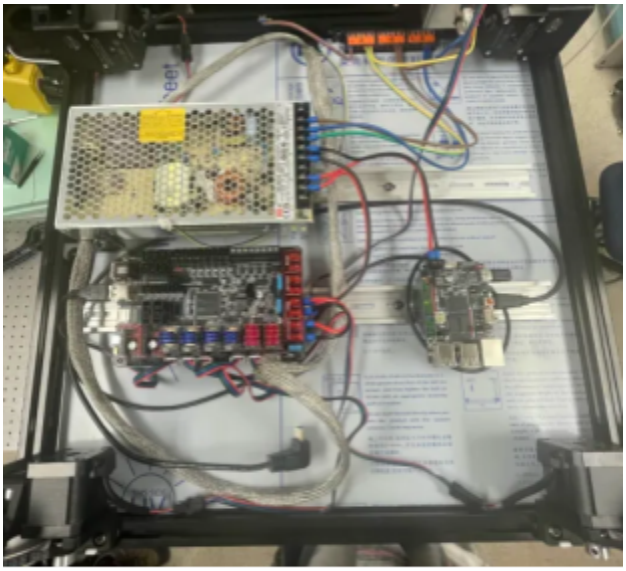


Figure 4.3.2-5: Electrical Bottom View

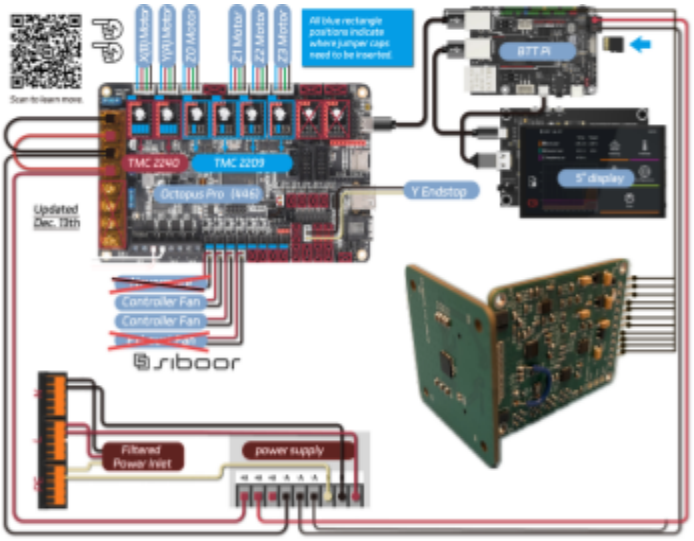


Figure 4.3.2-6: Wiring Diagram

Lzhikai. "Lzhikai/Siboor-Voron-2.4-AUG: Octopus Pro+btt Pi + CAN RP2040." *SIBOOR-Voron-2.4-AUG*, 2023, github.com/Lzhikai/SIBOOR-Voron-2.4-AUG.

Sensor

The most important part is the scanner head. It is a combination of two PCB's (printed circuit board). One, **Figure 4.3.2-7**, is known as the control board, which is connected to the electrical components on the bottom via wire to control the other PCB for sensing. The second PCB is known as the sensor head. This component is the actual sensing piece that will send and receive millimeter wave scans. **Figure 4.3.2-8** is the combination of those 2 PCBs that will be a part of the scanner head that will connect to the middle of the gantry system. The two of them combine to allow the control board to interpret the raw data returned from the sensing board then output it via SPI cabling through the DAQ. This is then stitched together using SAR algorithms to create a composite image.

Diving further in, the imaging device or radar is actually composed of several parts. The Digital Acquisitions (DAQ) device reads data out of the I+/- and Q+/-, and depending on the configuration this can trigger the radar sweep as well. The FTDI cable powers the radar, configures its registers, and commonly triggers the radar frequency sweep. We wound up switching from a National Instruments (NI) DAQ to using the Digilent Analog Discovery 2. This was due to a lack of Linux support from the NI DAQ.

Figure 4.3.2-7: Control Board

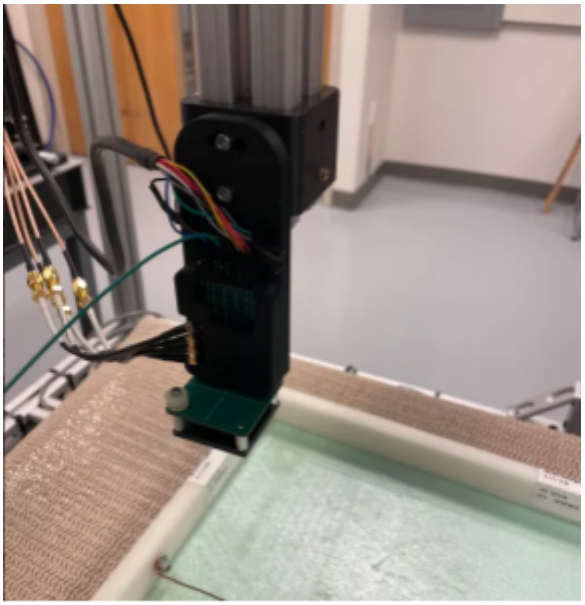


Figure 4.3.2-8: PCB Combination

2. Software

To ensure easy and convenient control of the scanner, we will implement a web application that runs directly on the Raspberry Pi, which will serve as the central control hub for the scanner.

The development of the interface and connection to the scanner included the implementation of html and css for the view of the application. The general functionality of how the front-end communicates to the backend and how the back-end communicates with the scanner is shown in **Figure 4.3.2-9** below

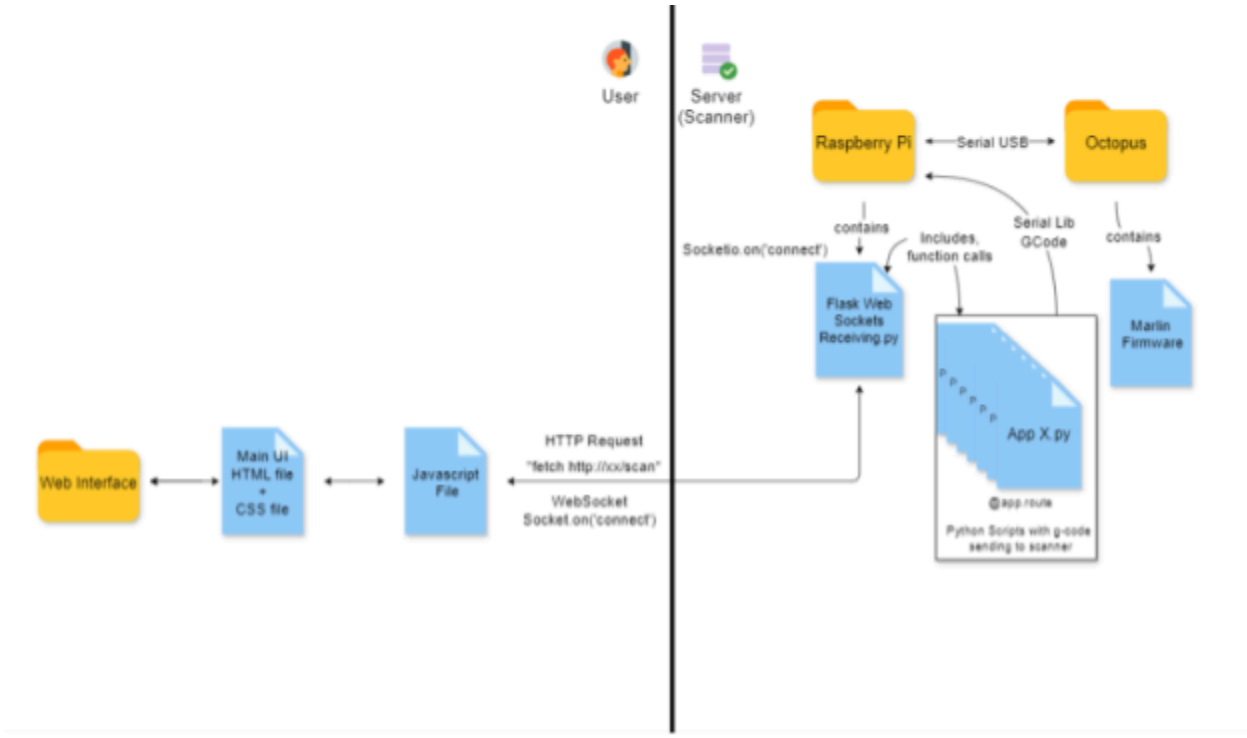


Figure 4.3.2-9: Software Architecure Diagram

Frontend-to-Backend

In the frontend, a javascript file, embedded in the web interface, handles all communication between the frontend and backend via:

- HTTP Post Requests: Used for actions such as:
    - Sending movement commands
    - Initiating a scan with user-defined parameters
    - Uploading .scan files

- HTTP Get Requests: Used for:
    - Retrieving scanner status
    - Querying scan results or system logs
    - Position updates
- Websocket communication: enables real-time bidirectional messaging between the frontend and backend, including
    - Live positional feedback during movement
    - Live status updates during scanning
    - Automatic saving on scan complete message
    - Notifications of system errors or completion events

These connections ensure the interface remains responsive and informative during all scanner operations. Below shows the frontend-backend communication via a POST request to move the sensor in a certain direction.

```python
@app.route('/move', methods=['POST'])
def move_axis_route():
    """Moves the sensor a specified relative distance along a specified axis. Updates SENSORLOCATION.

    Expects JSON data with keys: 'axis' (X, Y, or Z), 'dir' (positive or negative), and 'dist' (distance in mm).
    Example JSON payload: {"axis": "X", "dir": "positive", "dist": 10}
    Returns:
        JSON response with a message indicating the G-code command sent, or an error message.
    """
```

Figure 4.3.2-10: front-end POST operation (move)

Backend

The backend is built with Flask hosted on the Raspberry Pi. It is responsible for:

- Parsing and validating user inputs
- Generating appropriate G-code commands
- Sending commands over serial to the Octopus MCU
- Processing scan data and saving it to .scan files
- Trigger and manage scans
- Initiating SAR processing scripts

```javascript
async function moveAxis(axis, dir, dist) {
    const response = await fetch('/move', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ axis: axis, dir: dir, dist: dist })
    });

    const data = await response.json();
```

Figure 4.3.2-11: Backend POST operation (move)

The Flask server also serves the static frontend files and acts as a bridge between the user interface and the scanner hardware. As seen in the software diagram, we started with a series of separate files for the backend and wound up consolidating them all into one large file. In the end the MAVinator.py file contained all logic aside from the DAQ, radar, and scan pattern generation logic.

G-code + Hardware Control

All scanner motion and toolhead control are implemented using G-code, which is interpreted by the Marlin firmware on the Octopus MCU. Key functionalities include:

- Movement Commands: G0/G1 for linear motion, e.g., G1 X100 Y100 Z10 F3000
- Homing: G28 for homing all axes
- Scan Execution: Custom G-code sequences for moving in a grid and pausing for data acquisition and movements to complete.
- Digital I/O: Used to trigger scans and synchronize with the radar hardware

Each button press in the interface results in the corresponding G-code being sent via a serial connection to the controller, ensuring deterministic and repeatable behavior. **Figure 4.3.2-12** shows the terminal output of clicking the move button on the positive x with the input value of 30. It shows the corresponding g-code that is being sent to the motion controller.

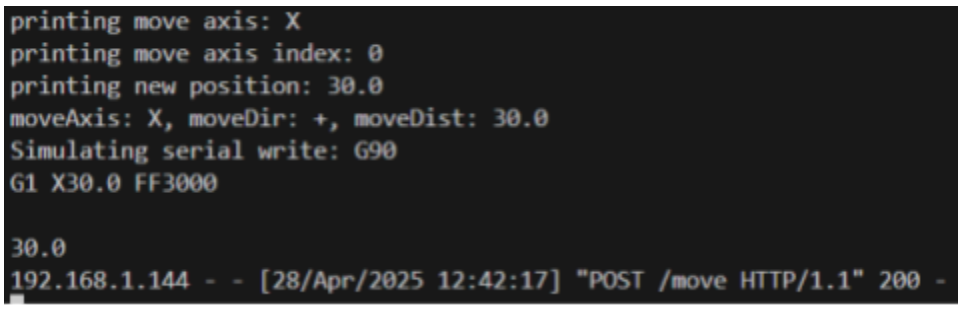


Figure 4.3.2-12: G-code output

In order to accommodate the new radar and DAQ we had to somewhat restructure how we had them wired together and mounted. The new radar has a very similar connection method and can be seen in **figure 4.3.2-13**. Still

using a FTDI cable we simply had to match the inputs and then connect the other end to our BTT Pi.

The DAQ did however require a junction with the purple (trigger) wire to one of its DIO pins in order for it to be triggered. The DAQ also required us to connect the I- and Q- lines to ground and simply read the positive outputs. The specific configuration of the Analogue Discovery 2 using the Digilent Waveforms python library was considerably more difficult to understand at first. Ultimately we needed to change the trigger position to be -½ the sample time as the trigger is meant to happen halfway through your reading by default. The DAQ is then connected back to the BTT Pi via USB.

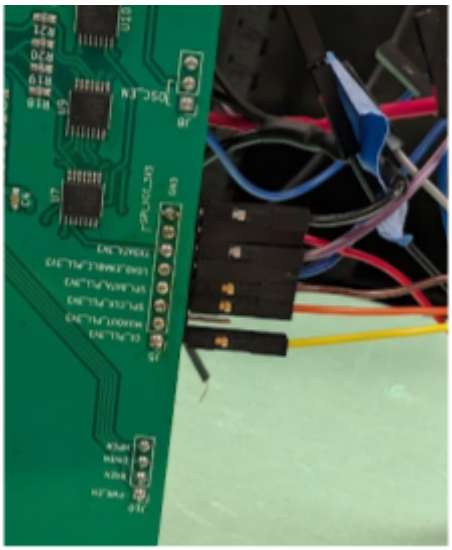


Figure 4.3.2-13

Visual Interface Layout

The web interface is divided into three main tabs:

- Move Tab:
    - Buttons for X, Y, Z movement
    - Homing controls
    - Live coordinate display
- Scan Tab:
    - Input fields for scan configuration (start/end positions, step size, delay)
    - “Start Scan” button
    - Live scan progress updates
    - Data preview plot
- SAR Tab:
    - Upload form for .scan files
    - SAR processing parameter inputs (e.g., resolution, clipping)
    - “Generate 3D SAR” button
    - Output display area

Each tab is designed with clarity and responsiveness in mind, utilizing CSS grids, semantic UI components, and error feedback alerts. **Figure 4.3.2-14** shows the first page that you see when you open the application. It shows the move page and all the proper functionality. Additionally it has the banner tab on the top to go to either the Scan or SAR tab.

Figure 4.3.2-14: Move Tab

Scanning

When on the Scan Tab, you can implement a scan and see the results in real time. As shown in **Figure 4.3.2-15**, in order to scan you need to input multiple different values for configuration such as: Name; X, Y and Z length and height; and X and Y step size. In addition to this you can click the “Configure” button which is optional and get a prompt shown in **figure 4.3.2-16** asking for specific frequency and time parameters. Once everything is configured, after clicking “Start Scan” you will start to see real time data being plotted.

Figure 4.3.2-15: Scan Page

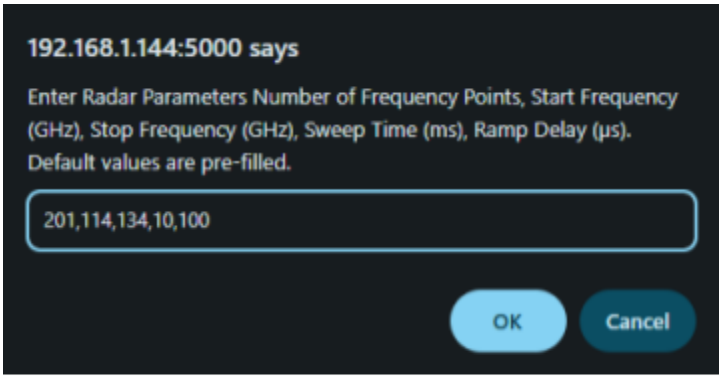


Figure 4.3.2-16: Configuration Prompt

Sar Processing

Upon completion of a scan, the .scan file can be saved and the user can upload any .scan file via the SAR tab and run SAR processing manually, by providing the Max Z depth and Z step size.

After clicking the Calculate 3D SAR button, The SAR Heatmap Slice output is displayed on the SAR tab as a downloadable image. The Below figure shows this page and how everything is laid out.

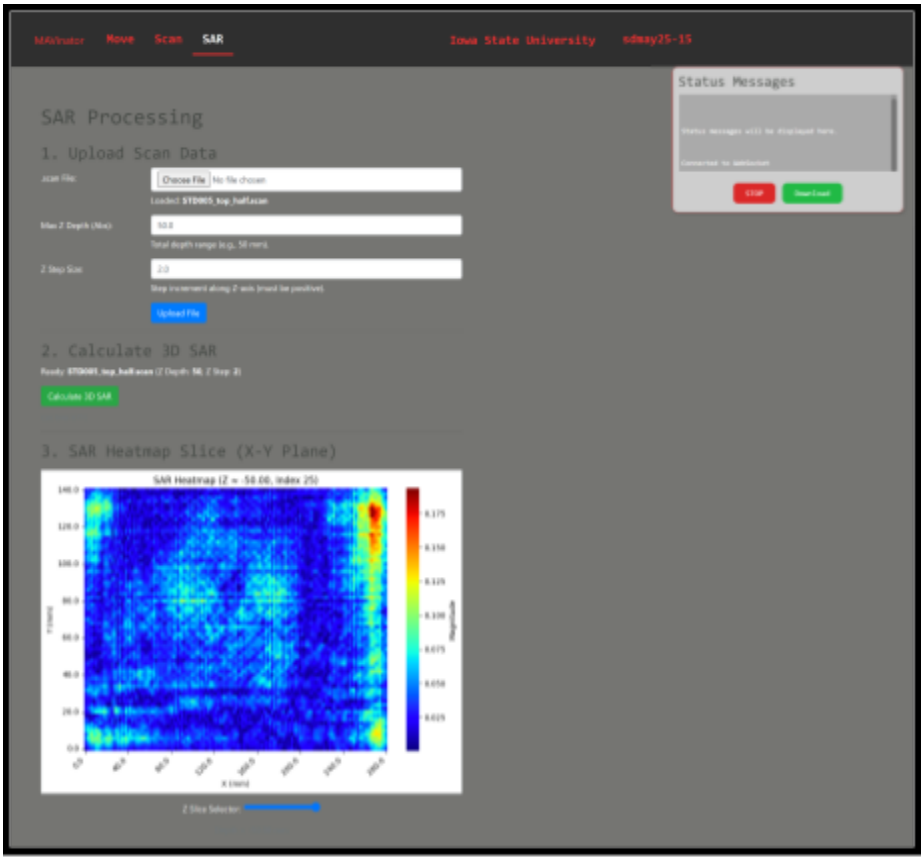


Figure 4.3.2-17: SAR tab

Error Handling and Logging

The system implements robust error handling across all layers:

- Frontend: displays error alerts for invalid inputs or disconnected devices
- Backend: logs all system events and errors to a text log, accessible via the interface

- Serial connection watchdog: detects dropped connections and auto-resets the controller

This is also implemented in the status bar in the top right corner of every tab as shown in the figures above. It outputs any error that occurs. This ensures maintainability and transparency for debugging.

4.2.3. Functionality

Our 3D scanning platform is designed to be intuitive and easy to operate, even for users without technical expertise. The system integrates a web application that controls the scanner remotely. Below, we outline how a typical user might interact with the system and how it would respond.

1. Setup and Initialization
    a. User Action: the user powers on the scanner and opens the web application on a browser.
    b. System response: The web application connects to the Raspberry Pi, which serves as the control hub and initializes communication with the scanner's hardware. The user is greeted with a welcome dashboard.
2. Manual Control Functionality
    a. User Action: The user manually moves the scanner using directional buttons or go to coordinates inputs and button.
    b. System Response: The scanner interprets these inputs as G-code commands, moves to the specified position, and updates the live coordinate display on the GUI.
3. Homing and Alignment
    a. User Action: the user selects the ‘home and align’ button to ensure that the scanner is in the correct starting position
    b. System Response: The Scanner head moves to its home position in the XYZ space, preparing for a scan
4. Running the Scan (Scan configuration:
    a. User Action: The user inputs configuration values and clicks the “Start Scan” button on the web application
    b. System Response: The scanner's stepper motors, connected by belts, begin moving the scanner head along the XYZ coordinates while the mm-wave imaging device collects data by emitting and receiving millimeter waves. This data is relayed back to the

Raspberry Pi for processing. The web application provides real-time feedback, displaying a progress bar and any relevant status updates.

5. SAR processing:
    a. User Action: Navigate to the SAR page and upload the file of the created .scan file. Set the max depth of the desired observance level and the step size. Lastly, click “Calculate SAR.”
    b. System Response: Once clicked, the system runs a Synthetic Aperture Radar (SAR) algorithm on the specified data to process and create 2D slices of the object, starting at 0 and goin down to the z depth specified by increments of the step size specified.
6. Viewing and Saving Results
    a. User Action: the user reviews the processed 2D map or image on the web application. The user can scroll down through the different layers of the data using the slider bar.
    b. System Response: the system displays the final 2D data.
7. Error Handling/Feedback
    a. User Action: causing potential error
    b. System Response: the interface will display an alert message. Logs are stored on the backend for troubleshooting
8. File Saving
    a. User Action: User clicks download button
    b. System Response: downloads most recent scan

4.3.4. Areas of Concern and Development

Our current design provides the basic functionality of a 3D scanning platform with millimeter-wave imaging capabilities. It meets the key requirements of a scanning volume of 300 x 300 x 300 mm and a user-friendly web-based interface. These features align well with the goal of delivering high-quality scans with ease of use for non-technical users.

While we are confident in the functionality and general design, we have some concerns. One concern regards the timing of the actual scan. Achieving a scan in a short period of time compared to the already existing scanners is a potential challenge. One other concern is about the complexity of the user interface. Although the web-based interface is intended to be user-friendly, ensuring that all users can operate it smoothly will require testing and may cause more of a challenge than expected.

Another concern is ensuring that scanner movement stays within defined bounds. Since the scanner operates using G-code instructions, improper or unchecked commands could result in movement beyond the physical limits of the frame, risking damage to the sensor head or mechanical components. Implementing reliable bounds checking in both firmware and software, and ensuring accurate homing behavior, is essential to maintain positional safety and repeatability.

Regarding the time requirement, one possible plan could be to experiment with the SAR processing algorithm or implement an adaptive scan that could reduce scan time as well. Regarding the user interface, to ensure that we create a user-friendly one, we can conduct usability tests with a sample group of users and collect feedback.

4.4. Technology Considerations

We have implemented a number of different technologies to bring the MAVinator to life.

- Voron Motion System

  The Voron motion system comes from a cannibalized Voron printer kit ordered specifically for this project. Voron's open-source licensing, modability, and large community make it an ideal technology to implement into our design. The alternative option would be to buy a three-axis motion stage/platform. While this would require a lot less assembly it would be exorbitantly expensive [1], hard to modify, and still require some setup physically or digitally.

  Other alternative options within the Voron family include any other version of Voron printer as the motion system. We selected Voron 2.4 with the core XYZ (floating gantry) due to the client's request and we agreed due to its higher theoretical top speed, availability of an example, and popularity.

  Within the Voron 2.4 motion system that we are using we could have selected other end-stops than the limit switches currently implemented. The alternatives there are using Hall effect sensors, or going for a sensorless homing process. We have shied away from doing so in the first phase of the design due to the complexity hall effect

sensors would add, because of the risks involved in sensorless homing, and because we have access to all of the limit switches that we need.

- 3D Printed Housing

  The Housing being 3D printed offers up a number of options for iterations and prototyping. The alternative in this case would be to have housing manufactured by a company such as PCBway through a more traditional method like CNC. While sending the designs off to have them manufactured could result in parts with higher durability the time trade-off is too severe. With 3D printing we can have a part made of PETG in a matter of hours or a day at most resulting in a much faster and more satisfying prototyping process. Additionally using a material like PETG or ABS can result in a part with more than enough strength.

- In-House Sensor Board

  The sensor and radar boards created in-house offer many advantages over other alternatives. They were designed specifically for the purpose of making millimeter wave scans. These boards have still required the soldering of all the surface mount and through hole components. The radar board is the more complicated of the two and is the one that creates the millimeter wave signal. Much testing has to be done in order to ensure that the board functions correctly as the main driver in the sensor. The second board contains the antenna and a biasing network to send out the signal generated by the radar board and then capture the return signal to be processed. The frequency sweep is set up to be triggered by a digital signal from the FTDI cable.

  Having the boards designed in-house gives us great access for any questions or concerns we have about the operation of the boards. Several of the workers at the lab have experience with testing and issues associated with them so we will be able to use their expertise to help us diagnose issues along the way.

- FTDI cable

  The FTDI cable is basically a usb to digital pin-out adapter. We utilize this to program the registers that tell the sensor what frequencies to sweep over and other KWARG(see gitlab) based parameters. Last

minute we also switched from using the National Instrument to trigger the radar sweep to using one of the FTDI cables 3v digital I/O pin. In order to ensure the sweep is triggered at the same time as the measurement Luke made a spliced cable that took that signal and branched off two ways to the DAQ and radar. This is plugged into the Pi.

- Digilent Analog Discovery 2 (USB oscilloscope)

  The Digilent Analog Discovery 2 (AD2) was a last minute consideration, but a very necessary one. This little device packs a lot of functionality in a small form factor. We utilized its python SDK after familiarizing ourselves with the WaveForms software user interface. The wide range of functionality made it a little trickier to figure out how exactly to configure the device as a DAQ for our purposes. We hardwired one of the digital I/O pins on the AD2 to listen for a trigger that comes from the FTDI cable. This starts our preconfigured recording process until the buffer is filled up at which time we reap the data.

- Flask python webserver

  In the MAVinator project, Flask serves as the backbone of our web-based user interface. We've implemented Flask on the Raspberry Pi to act as a server, managing communication between the frontend web application and the underlying hardware control system. Flask handles all requests from the user interface. Such as movement commands, and translates them into G-code instructions for the Octopus MCU. In the case of a scan it manages the thread that generates the g-code and triggers the radar system at each point.

  Furthermore, Flask handles the processes for scan data, saves it to files, and triggers SAR processing algorithms. It also facilitates real-time updates to the user interface through websockets, providing live feedback on scanner status and position. Essentially, Flask acts as a nervous system, orchestrating user interactions, hardware control, and data processing within our MAVinator system.

5. Testing

5.1. Unit Testing

The MAVinator is composed of two main physical parts that need to undergo testing: the Voron itself and the two circuit boards. The following sections will discuss the testing of each component. After the completion of the previous circuit board assembly we had to swap to a new radar which was tested for us and is verified as satisfactory. In the second half of our project we focused most of our testing on the code we created.

5.1.1. Voron Build

The Voron based portion of the scanner requires testing, and will occur in three primary phases: Electronics smoke test, Basic motion testing, and Advanced Motion testing. This phased testing will help to mitigate some risks, risks of electronic component failure, risk of mechanical damage to motion system components due to “dumb” motion, and risk of damage to the millimeter wavelength sensor.

The first test upon completed construction consists of powering on the printer with protection, our smoke test if you will. While the scanner is plugged into a surge protector that has protection for shorts we will turn on the power switch and inspect the printer with power on for 1 minute or until we see something of concern. If there is a short due to component failure, the surge strip should trigger and prevent catastrophic failure.

With a successful smoke test we begin the testing of the functionality of the motion system with simple movements. After powering on, using a provided test script we will ensure that the gauntry shuttle can move in the X, Y, and Z directions both positive and negative. This test will not test the outer or inner bounds, just short movements in all directions. If the distances, and directions are correct and as expected then we will move onto the final.

Lastly there is the advanced motion test. This test consists of implementing automated paths and making sure they operate as intended. The automation that will be tested is “homing” the gauntry shuttle. In this motion the shuttle is moved to X = 0, Y = 0, and Z = 350 with each motion ending when the shuttle triggers an end stop. The first position moved to should be the X, then

the Y, and then the Z in order to properly trigger the third end stop. With that test successful and positional accuracy established, we are ready to attach the sensor.

5.1.2. Circuit Boards

The circuit boards required extensive testing as there are several opportunities for errors to occur in the soldering process. Resistors or capacitors could stand up in the reflow oven, integrated circuits could be placed with the wrong orientation, or components could be shorted to ground. Testing began with the more populated board which will henceforth be referred to as the control board. The second board will be called the radar board.

Testing of the control board began by visual inspection after soldering. Immediately a couple of components were discovered that were not soldered down correctly. The second step of the testing phase was ensuring that nothing was connected to ground that wasn't supposed to be. This is done by using a multimeter and tapping soldering connections along the board to check if they are grounded. Again, several connection points were discovered to be incorrectly soldered. Those points were fixed, and testing continued with path tracing. Path tracing involves following connected paths from the power source back to where the power is needed to make sure there are no breaks in the path. This test showed no issues and we moved on to the next major test: connecting the control board to power.

The main purpose of this test is to check if the correct voltages are showing up at every node. The board was tested using an FTDI cable to supply the voltage. This test raised a flag which meant that something was awry. After doing several tests with the integrated circuits and shift registers, then taking them off completely and the error remained. This showed that the issue was not related to those components. After some more investigation, we discovered that the oscillator on the power board was shorted, but it was covered up by a jumper wire not included in the original design which is why it was originally overlooked. Once we took the oscillator off, cleaned up the solder, and placed it back on the board, the power was much closer to accurate, and we moved on to the radar board.

The radar board was much more simple to test as there are much less components. The same tests were done on this board as on the control board. The test for shorts came up with a few issues, but those were solved and no other problems were detected.

We followed up with testing the two boards when placed together. When checking the voltage levels at critical nodes, we noticed a lower voltage level than expected, but continued to try and program the PLL on the board using the FTDI cable. We discovered the board was not beginning programmed so we traced each critical signal from the FTDI cable to the PLL using a picoscope and discovered through this that a resistor was not fully soldered and the PLL got rotated the second time we were putting it on the board so the pins were misaligned. After fixing these two issues, the board passed all of the tests needed before we attach it to a DAQ and observe the output.

The final test was to ensure proper functionality. We attached the IF_Q+, IF_Q-. IF_I+, and IF_I- cables to the control board and set up the PCB system on a scanner which allows us to read the data. The scanner in this test will be replaced by the Voron in the final product. A piece of foam was scanned with nine rubber plugs located at different depths throughout the foam. The data was then filtered and had synthetic aperture radar(SAR) applied to it which created two and three-dimensional images and those images came back great, which means the PCB system is ready to be attached to the Voron.

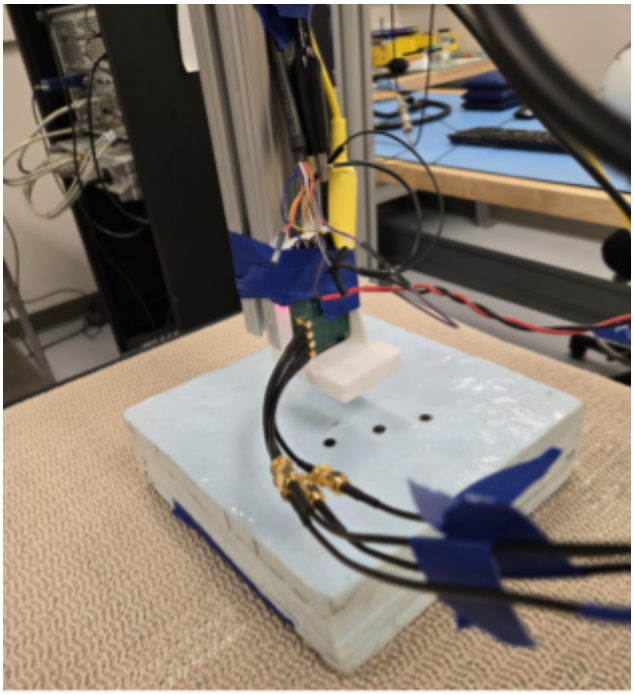


Figure 5.1.2-1: Test Scan Setup

5.2. Interface Testing

This section will outline the testing procedures for user-entered parameters, including inputting unexpected, excessive, and mismatched data. Additionally, it will detail regular use case testing, which has revealed that frontend data does not persist on page change. This section will also cover the testing of SAR algorithms using established scan files and their comparison to a MATLAB script implementation.

5.2.1. Backend API Testing

G-code testing

- Boundary Testing:
    - Ensuring that movement commands (G0/G1) respect the defined limits of the scanner's x and y axes and does not attempt to move beyond them, preventing physical damage. This was done by using Pronterface to issue g-code that exceeds known bounds directly. The power switch on the back of the scanner served as an emergency stop.
- Z-Axis Safety:

    - Implementing and testing restrictions on Z-axis movement to prevent the sensor head from colliding with the scanned object or the base, incorporating both software and hardware (limit switch) checks. This was tested after we had the UI implemented, ensuring both the scan and motion controls wouldn't go below our defined threshold of 100mm with the new sensor.
- Homing Accuracy:
    - Verifying that the G28 homing command consistently and accurately returns the scanner head to its designated home position, ensuring reliable positional reference for scans. This was done using the initial Pronterface testing to issue a G28 over a serial connection.
- Scan Pattern Validation:
    - Testing the custom G-code sequences that define scan patterns (e.g., grid movement, data acquisition pauses) to confirm they execute as intended and produce the desired scanning behavior. This was tested after the backend was more developed, using NCview G-code simulator.
- Digital I/O Synchronization:
    - Thoroughly testing the digital input/output signals used to trigger scans and synchronize the scanner's movement with the radar hardware, ensuring seamless data acquisition during scans. This was tested after the swap to the new radar and the backend was almost completely developed. To test we ran scans and ensured that the data readings were occurring within the allocated pause between each movement, printing to the console when a sample ended and then when the pause ended.

SAR

- Data Extraction:
    - Testing to ensure the data is imported in the correct manner and that the array sizes of the data are correct. Also testing to verify that the real and imaginary pairs get placed together properly.
- Data Processing Comparison:
    - Verifying that the data processing scripts result in the same data as the MATLAB processing scripts currently used by the CNDE to

perform SAR calculations. Also need to confirm that the arrays are returned with the correct dimensions.

- Data Heatmap Plotting:
    - Test the plotting feature to ensure that the data is being plotted on the correct axis with the correct magnitude, and verify those results using the pre-existing MATLAB scripts.
    - We also tested to ensure that the slider bar allows proper scrolling through the slices of the scan, taking the correct step sizes, and not taking too long to update the interface.

DAQ Testing

- Waveforms GUI and Data Comparison
    - The AD2 was initially connected to a laptop and controlled using the Waveforms GUI. Data was collected and compared against recent scans from the previous DAQ to ensure data consistency and quality. This visual comparison allowed us to quickly see that half of the buffer was empty at that trigger position. This resulted in half of our sample being lost due to the default trigger position being set to start with the buffer half full already.
- Object Manipulation and Sample Rate Verification
    - To verify the sample rate and overall responsiveness of the AD2, we placed various objects in front of the sensor and manipulated them while observing the real-time data stream in the Waveforms GUI. This dynamic testing helped us confirm that the AD2 was capturing data at the expected rate and accurately reflecting changes in the environment.
- Integration with Backend and Firmware
    - Once initial testing in the Waveforms GUI was complete, we integrated the AD2 with the Raspberry Pi backend. This involved writing custom Python scripts to control the AD2's settings and trigger data acquisition in sync with the scanner's movements. Thorough testing was conducted to ensure seamless communication and data transfer between all components.

5.2.2. User Interface Testing

The web-based user interface was built using HTML, CSS and JavaScript and tested iteratively through development. Testing focused on usability, responsiveness , communication with the backend and visual clarity

- HTML
    - verified structural layout and element accessibility across browsers
    - ensured all forms, buttons and input fields rendered properly and were mapped to functional javascript actions
- STYLING (CSS)
    - Ensured responsiveness using flexbox/grid layouts and checked for any layout breaking under user interaction.
    - Confirmed color themes matched CNDE's aesthetic preferences and were accessible for low-vision users.
- JAVASCRIPT (connection to backend)
    - Ensured all frontend buttons triggered the correct HTTP POST or WebSocket events.
    - Tested error handling (e.g., empty input fields, out-of-bounds values) and validated feedback to the user.

5.2.3. Hardware Interface Testing

DAQ Hardware interface

The DAQ hardware interface did not require much testing as much of it is very much lab ready but not field ready. The connections were tested to make sure they had continuity and were taped to ensure minimal interference. The DAQ hardware itself was tested by comparing its output signal to a known input signal using WaveForms SDK.

Radar hardware interface

The hardware interface required a longer, less involved test duration to ensure durability of the print. This particular print had a lot of internal stresses that resulted in some prints breaking themselves once cooled. Additionally we test for alignment by comparing the final connection points to the PCB it will mount.

5.3. Integration Testing

Initially, we conducted integration testing by issuing serial G-code commands from the raspberry Pi to the marlin based control board, ensuring that the integration of the two did not break either. This then allowed us to confidently test the integration of the hardware mount for the sensor with the Voron motion system, ensuring that neither impeded the function of either. We then repeated these initial tests with the new sensor mount for the new sensor. Following the initial tests, we proceeded with comprehensive testing of the new sensor integrated with the new DAQ. During his phase performing complete scans of various objects and comparing the results to known good scans of the same subjects verifies the successful integration. This comparative analysis will enable us to assess the accuracy and reliability of the new sensor, ensuring that it meets the required standards for millimeter-wave imaging.

By conducting thorough testing and validation at each stage of development, we aim to ensure that the MAVinator delivers accurate and reliable 3D scanning results, meeting the needs of our users and providing a valuable tool for millimeter-wave imaging applications.

5.4. System Testing

System testing encompases using the system as a user would from start to finish. The system's web interface should be able to easily navigate to all the control tabs; move, scan, and SAR. The controls available on each page should be immediately apparent. The system should be connected to the backend websockets immediately on each page. Critically, the system will be able to run a scan with minimal interaction. This includes moving the sensor head to initial position, and then pausing, sampling, moving and repeating. When this completes a standard .scan file is saved to the system upon success. This scan file can then be downloaded and uploaded to the SAR page for image generation. The SAR image is then easy to navigate through the layers of. Throughout all of this the user should be notified of errors but they should not impede the operation of the system overall.

5.5. Regression Testing

Regression testing started once we started work on the firmware which occurred at the start of the second semester. Constant regression testing ensures that new code changes don't break existing functionality. Initially performing tests on the printer to ensure that our modifications to the firmware did not reduce functionality or reliability. More typically, post-merge testing after we started writing the Flask code was crucial to catch integration issues early. This was to prevent unexpected problems due to dependencies, conflicts, or other differences. These post-merge issues are often addressed with small, targeted code changes like bug fixes, configuration adjustments, and dependency updates.

5.6. Acceptance Testing

The design requirements we were presented with include having a motion volume of 300mm x 300mm x 300mm or larger, an accuracy of 0.5 mm, a graphical user interface, the ability to perform a scan on a uniform cartesian grid, process the data using the SAR algorithm, and display the results. To ensure our product met all of these requirements, we gave our product to our client, Dr. Tayeb, and let him analyze it. He mentioned that we have created the minimal viable product that can now be pivoted to the exact desire of the users, which is exactly how Apply phones started.

5.7. User Testing

We have managed to test once with our primary user, Dr. Tayeb. Though we have not done all the tests that we would like we did get a number of them with one user. We asked Dr. Tayeb to move the printer as he desired using the available movement commands on the move page. The user was asked to home the printer and then navigate to the scan and SAR pages. We followed up the navigation test with a scan test in which our user confirmed our readings looked good. We would like to have the user save a .scan file and reupload it for SAR processing. The last test performed was a test of the SAR processing using a known good scan, asking Dr. Tayeb to navigate through layers of the SAR image.

5.8. Results

5.8.1. First PCB

The following results came from the PCB testing. These results come from a scan of a piece of foam with nine rubber plugs dispersed at three different levels within. The plugs can be clearly seen in the images which was the desired outcome. This testing was done with our original PCB, but we were required to change the PCB used for the final product due to extenuating circumstances.

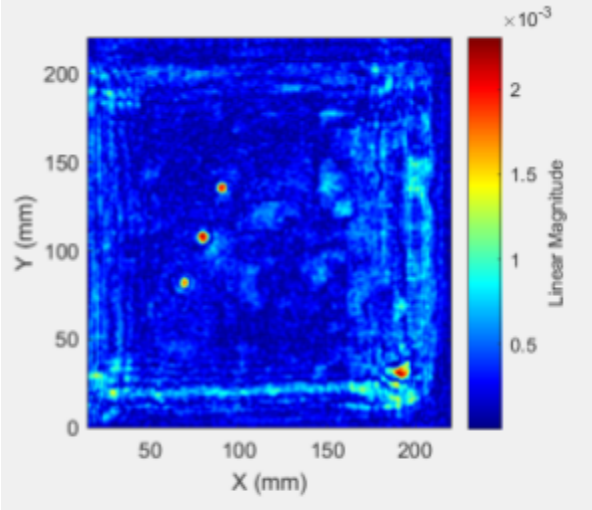


Figure 5.8.1-1: Top Three Rubber Plugs

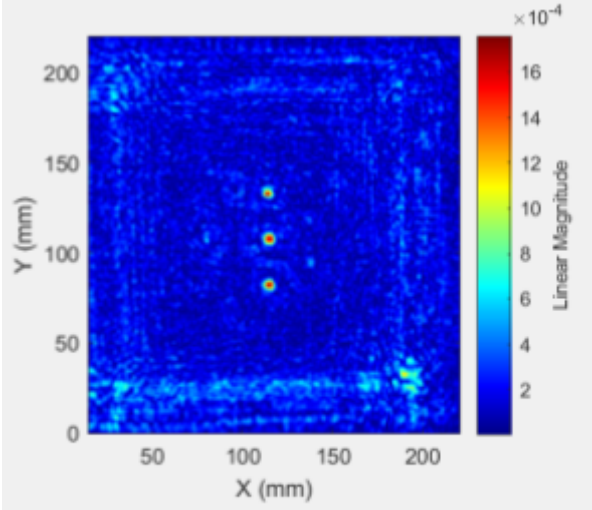


Figure 5.8.1-2: Middle Three Rubber Plugs

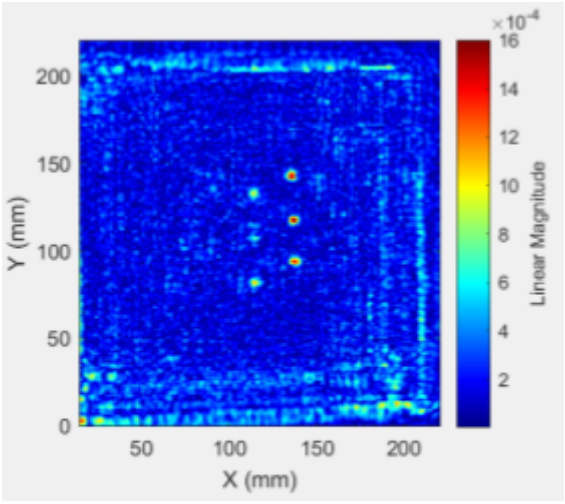


Figure 5.8.1-3: Bottom Three Rubber Plugs

5.8.2. SAR Processing

To verify the results of the SAR processing of our web application, the results were compared to the accepted results of the MATLAB program created by Matthew Dvorsky. As can be seen in Fig. (5.8.2.-1) and Fig. (5.8.2.-2), the results are extremely similar, and the defect can be clearly seen in the center of both images. The difference between the two is the scaling of the axis. The MATLAB has interactive scaling where the x and y axis can change total length whereas the web application always displays the SAR results in a square plot.

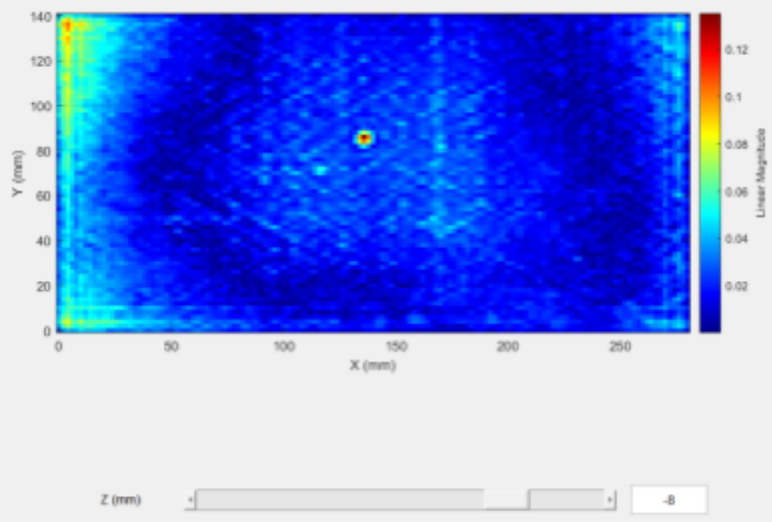


Figure 5.8.2-1: SAR Processing using MATLAB

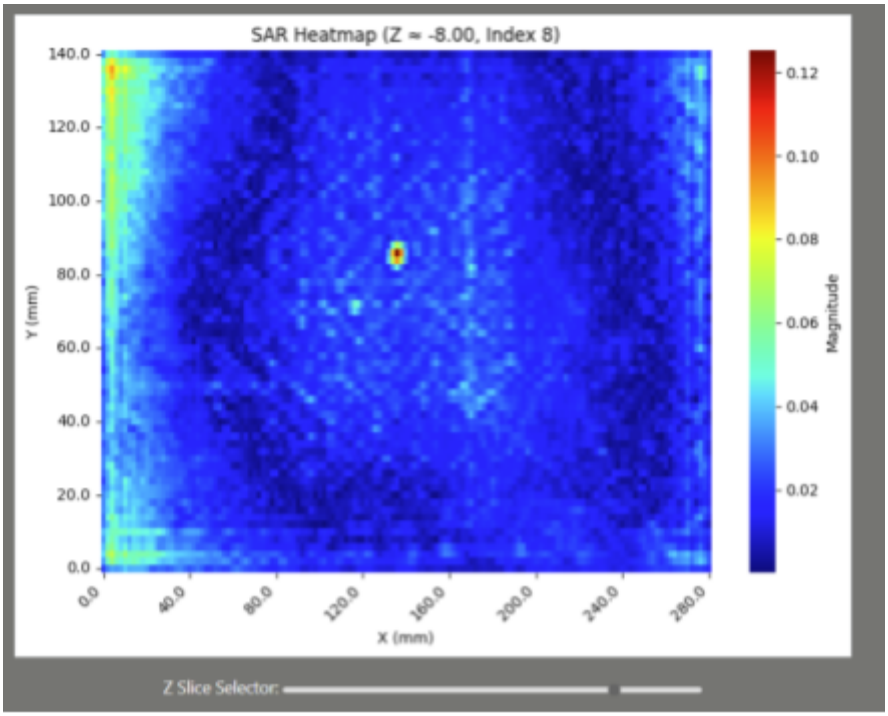


Figure 5.8.2-2: SAR Processing using Web App

5.8.3. DAQ

When we replaced the DAQ we needed to test the measurement processes we configured the DAQ for. We did this by comparing the data received to that of known working radar output using our real time data visualization. From these results we saw the expected curves, in the images the surface measured is different.

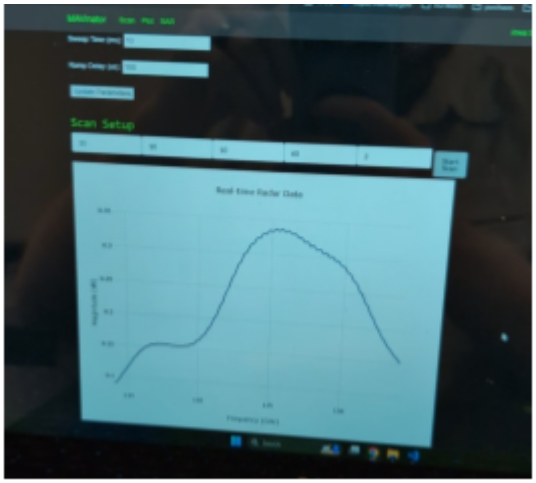


Figure 5.8.3-1: Control DAQ Data

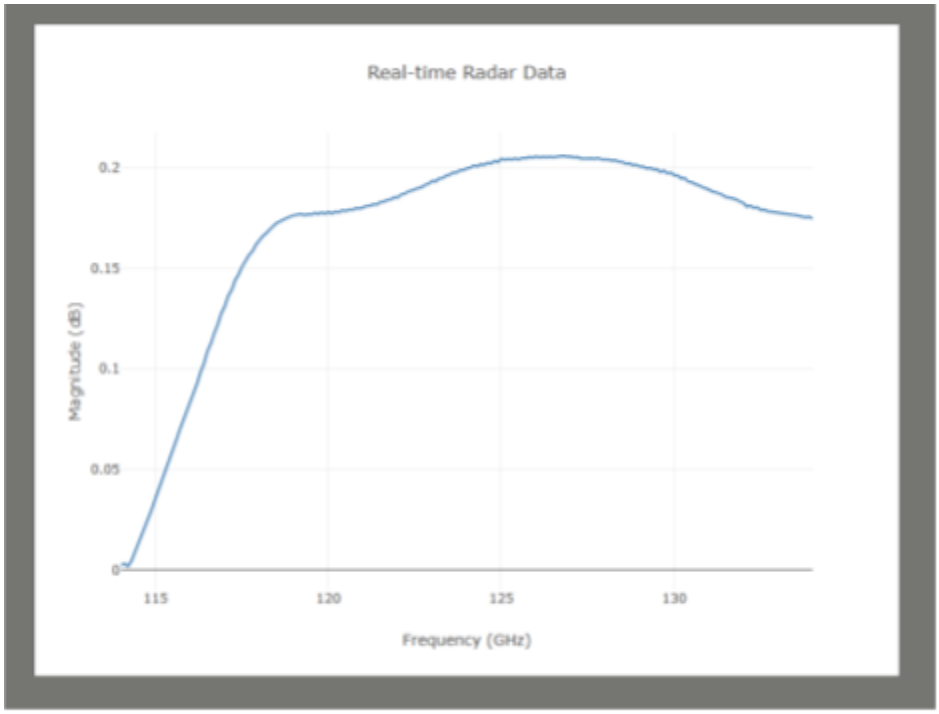


FIgure 5.8.3-2: New DAQ response

5.8.4. System Testing (scanning)

These tests, while not complete, informed us of the basic next steps and shortcomings we face. We conducted the simplest scan possible, a 20x20mm scan using default radar parameters. This gave us very good results as everything performed as expected. The next tests will consist of non-square test grids in both directions, higher resolution scans, and taking a higher resolution scan of control objects that have known good and easily interpretable scans.

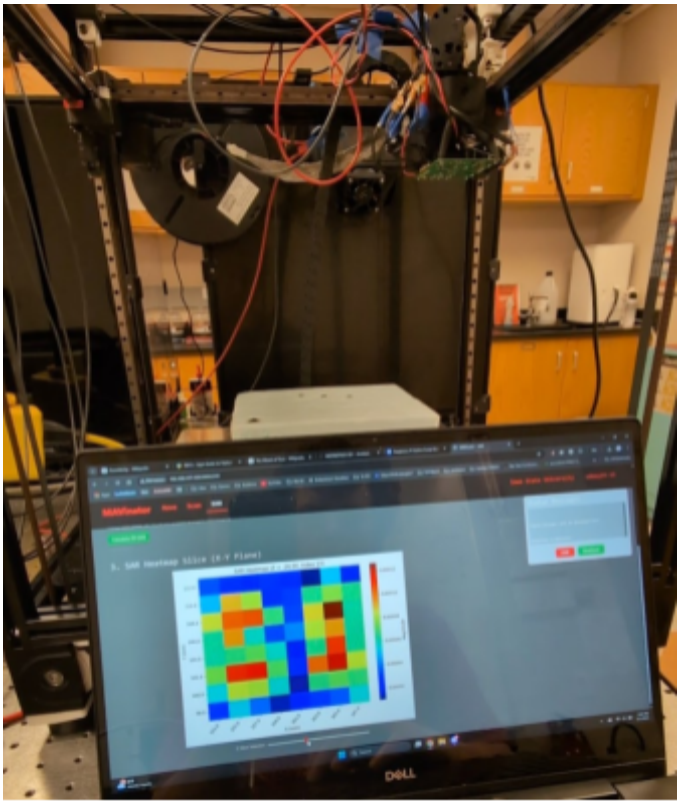


Figure 5.8.4-1: System test post SAR

5.8.5. User Testing

We had two users, Dr. Tayeb and Matthew Dvorsky, test the MAVinator. The MAVinator performed adequately, but both of them gave us valuable criticisms. The most impactful criticism came from Matt, we should make our scans start relative to the position of the sensor head, not centered on the bed every time. Matt also noted that it would be more useful to have our Go-To function automatically enter the sensor's current location so that you can just modify one of the three coordinates. Dr. Tayeb also noted that the SAR page should give the user the option to adjust the Z-step size without re-uploading the file. On the same page for SAR processing he theorized that we should be able to pull the max Z-depth out from the .scan file information. With these results sdmay15 can move forward confident that our next changes will bring us closer to the desired finished product. There is still much user and acceptance testing needing to be performed.

6. Implementation

The implementation of the complete product was done over two semesters and broken into two main stages, with the first being hardware and done over the course of the first semester and the second being the software and being done over the course of the second semester

The implementation of the build of all the “hardware” has been completed in four different stages: mechanical assembly, electronics integration, sensor and housing, and end stops.

The implementation of the build of all the “software” has been completed in five different stages: HTML requests, websockets, G-code generation, SAR processing, and DAQ and Radar Classes.

While Section 4. Design provides an in-depth look at the technical details, this section offers an overview of the work completed this semester.

6.1 Hardware

6.1.1 Mechanical Assembly

Thus far, the majority of the mechanical assembly for the MAVinator scanner has been completed. We began by constructing the frame using a Voron 2.4R2 printer kit as our foundation. This open-source motion platform provided a base. Following the Voron build guide, we first assembled the frame, ensuring that it was square. The gantry system, composed of linear rails, belts, idlers, and stepper motors, was then integrated to enable precise three-axis (X, Y, and Z) movement. The mechanical assembly now closely resembles a high-precision 3D printer, but repurposed for millimeter-wave scanning.

During this assembly phase, we focused heavily on proper alignment and tensioning. The drive belts were carefully tensioned to ensure smooth, backlash-free travel, and linear rails were checked for parallelism to meet our 0.5 mm positional accuracy requirement. Although this process involved significant iteration—tightening, loosening, realigning—we have achieved a stable, rigid motion platform capable of consistent, repeatable movement.

6.1.2. Electronics Integration

Following the mechanical build, we began integrating the electronic components. We mounted the main controller board (Octopus MCU), power supply, stepper drivers, and Raspberry Pi onto a lower deck beneath the scanning platform. Each stepper motor has been wired into the motor drivers, and initial continuity checks have confirmed that all wiring connections are correct and secure.

Before adding sensors and end stops, we ran a basic power-up test to verify the correct voltage outputs from the power supply and confirm that the controller board powered up without issue. Preliminary tests show that the motors can be energized and that no electrical shorts or grounding issues were present.

6.1.3. Sensor and Housing

After completing the soldering and testing of the PCBs, we began to implement it into the Voron. This required us to design a housing for the sensor (PCBs put together) that would fix it to the Voron extruder. We used an existing design for the housing as a base template for our design. Modifications were necessary to attach it to the pre-existing mount on the Voron and to hold the radar part of the scanner more firmly in place as it will experience some vibrations as it moves around the scanning platform.

6.1.4. End Stops

The Voron kit we were supplied with did not come with the necessary end stop parts to ensure that the extruder would not travel too far in any direction. The Voron step file model contained many different mount versions, so we simply selected the right one, then printed our own pieces. The kit did come with the Y and Y end stops but the wires were too short for our implementation so we redid those to make them the needed length and routed them to the mount we printed.

The Z-axis end stop did not come with the kit. The CNDE lab had two more already assembled Voron printers that were no longer using their Z-axis end stops so we were able to salvage one of those end stops for our purposes. Typically this end stop is located at the bottom of the printer, but for our application, we needed it at the top. It is not designed for that, though, so we

created another 3D printed part that would hit the end stop when the extruder travelled all the way to the top of the Voron frame in the home position.

6.2. Software

6.2.1. HTML Requests

HTML requests serve as a bridge between the frontend and the backend. When a user clicks a button (e.g., "Start Scan", "Move Axis", or "Home"), the HTML element (button) is connected to a corresponding JavaScript function. This JavaScript function triggers an HTTP request to the backend, typically using the fetch API with a .text or .json response type, depending on the desired format. Once the backend receives the request, it routes the request to an appropriate function, which processes the request based on the functionality specified (e.g., starting a scan, moving an axis, or homing). The backend then communicates with the motion controller, sending the corresponding G-code to perform the requested operation.

6.2.2. Websockets

Websockets are used lightly in the design of the backend, primarily for scans and data processing. Flask-socketio is used to send initial connection messages, disconnect messages, live errors as they happen, and scan status information. When a start scan message is broadcast the backend websocket listener starts a new thread to generate the g-code and starts sending the commands via serial. Scan started is broadcast back over the websocket connection as well as an eventual scan completed message. These along with other error messages broadcast via websockets are displayed in the status box visible on all pages. We had an implementation of live websocket position updates but this was not as efficient as keeping track of the position with each movement command.

6.2.3. Gcode Generation

The G-code generation for scanning is done significantly differently from G-code generated for movement commands. Movement commands are hard coded with a preceding relative move mode command, G91. This is then followed by a hard coded G0 command with the delta relative to the scanner’s current position. Alternatively this is done with a G0 command

directly for go-to-position style movement commands. For both operations the sensor location is updated with each movement.

In the backend the pattern G-code generation is handled in a somewhat modular manner. This modularity could be enhanced with a base python class that other scan pattern classes would overwrite. In our case it is simply handled within a function that is broken off into its own file. This function takes in the object dimensions, step sizes, and z-height, then generates a snake-wise pattern starting from the front right corner of the scan plate. When the dimensions are entered in the front end they are verified as easily divisible by the step sizes or not and the user is given options to automatically correct the discrepancy. Once the pattern is generated it is stored in a python list of G-code commands then each one is processed and the sensor location is updated.

6.2.4 Scan File Saving

Scan file saving is done by saving operational data as well as measured data at each scan point. The operational data includes the frequency range that the scanner is scanning each point at. These values are stored in an array that is cleared and then populated each time a scan is initiated by pulling it from the sensor itself. The frequency range only needed to be calculated once per scan and not at each point like the other data. The other operational data that is taken at each scan point is the coordinates that the scan took place at, so the X, Y, and Z coordinates. Each iteration would store an array of the coordinates to another array for processing later. Lastly, the measured data coming out of the radar needed to be stored in an array as well to correspond with each coordinate stored. After a scan is complete and the data is stored in their respective arrays, they are passed to the export_scan function to be exported into a .scan file. The export_scan function takes the filename given by the user in the scan page, the coordinate points, the measured data points, as well as the frequency range used by the radar to generate the .scan file. The export_scan function was derived from a given matlab script that we converted to python for use in our scanner.

6.2.5. SAR Processing

The purpose of the SAR processing is to return images of the collected data that is capable for a human to visually analyze. It takes the returned

S-parameters, turns the data into the time domain, and returns the magnitude at each data point taken. For this project, we were provided with working MATLAB code that needed to be converted to python code to use in the web application. Five total files needed to be translated: a file to import the data, a file to perform 2D processing on the data, one to perform 3D processing, one to create 3D SAR data in case you have bistatic separation, and one to calculate the gaussian of the antenna. For the current implementation of our project, only the scan importing and 3D processing are necessary, but the others were also translated in case of future work.

Import Scan

The import scan file takes an input of a .scan file and returns the x, y, frequencies, data, and the header. The imported file has all of the real and imaginary data separated in one long list, so it is necessary to group the pairs back together. That data then needs to be assigned to their respective x and y coordinates. This proved more difficult than expected because the 'reshape' function in MATLAB does not work the same as the 'reshape' function in the NumPy library. This made it necessary to manually assign the data to the correct coordinate pairs. After that is complete, the data is transposed to the correct dimensions.

SAR Processing 3D

This file does all of the necessary processing on the data to make it easily understandable by the user. It can take several arguments, with the necessary ones being the data, x, y, and z coordinates, and frequency coordinates, and the unnecessary but more commonly used ones being the zero pad percent and the option of removing averaging. Some initial preprocessing is done on the data to apply the zero padding and calculating the wavenumber.

The data with the zero padding applied then has 2-dimensional Fast Fourier Transform (FFT) applied to it to turn it into time domain data. This allows us to 'look into' the material being scanned by knowing how long it took the wave to reflect back to the antenna. That data is then divided by the wavenumber. Next, the data is mapped to the z-depth values. Some more data processing is done to get the image to appear correctly, and finally the inverse FFT is calculated to finalize the plotting of the data.

6.2.6 DAQ and Radar Classes

The DAQ and Radar classes, much like the g-code pattern generation, were designed in a somewhat modular manner. Writing these in a more modular fashion with parent classes that could be implemented or overwritten would have aided this project when it came time to switch DAQ's and sensors. Regardless, our implementation was done through direct addition of classes (ADF and Simulated) to the Radar class and adding functionality to the DAQ/FTDI classes to support a Digilent DAQ's software development kit.

The switching of the radar itself did not impose many changes, but the switching of the DAQ did actually require a number of tricks. The first trick to using the Digilent Waveforms SDK was having its specific dll library installed on the Pi. The second was setting the digital I/O 1 pin to be a trigger in order to ensure timing accuracy regardless of computer constraints. Lastly the third trick was moving the trigger position to be half of the sample count multiplied by the sample rate.

6.3. Design Analysis

Our implemented design works to the standard that we were given. We can successfully move the sensor through the use of the web application. The tracking of the sensor head through sliders works well, but the speed does not directly match. We have proven that we can run a scan that collects accurate data using the new sensor we received and DAQ. We are then able to save that data in a .scan file to the user's device. That .scan file can then be uploaded to the web application and have SAR processing performed on the data to a specified depth with a given step size. That data is then plotted in a heatmap image that the user can visually decipher. The web application also has a more modern appearance which was desired by our client. This all works because we set out with a definitive goal in mind, and worked diligently until we got the results we were looking for.

One part that does not work quite as well as expected is the SAR image displays an entirely green heatmap at a depth of 0, which is not expected. Fortunately, that data is not used as the sensor is never positioned directly on

top of the material being scanned so this is not a major issue. All of the other data is plotted correctly.

Another shortcoming we noticed in our design was that when each page is refreshed the data entered is lost. This is not a major problem as most are simple values, though for repeated activities over a longer work session may get repetitive.

7. Ethics and Professional Responsibility

In designing and developing the MAVinator scanner, our team recognizes that engineering ethics and professional responsibility extend beyond technical correctness. Ethical conduct involves considering how our work affects users, the environment, society, and compliance with professional standards. We aim to uphold the highest ethical principles, ensuring safe, beneficial, and equitable outcomes.

7.1. Areas of Professional Responsibility/Codes of Ethics

Area of responsibility	Definition	IEEE	Team interaction
Willingness to learn and improve	Having an open mind and active desire to gain new knowledge and skills, constantly seeking ways to enhance your performance or abilities in any given situation	To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, to be honest and realistic in stating claims or estimates based on available data, and to credit properly the contributions of others;	The team has constantly sought out feedback internally, from our advisor, and at certain points from external sources as well. Always giving great care to the feedback received and finding ways to take it into account.

7.1.1. Area in Which the Team is Performing Well:

One area in which we are doing well is communication honesty. This is defined as Perform work of high quality, integrity, timeliness, and professional competence. The team has had honest and transparent communication with each other and has assisted in high quality work. By maintaining open lines of

communication with our mentor, we can acknowledge when we are behind schedule and also actively seek feedback on any issues we have. This approach enables us to meet professional standards in timeliness, competence, and overall project integrity

7.1.2. Area in Which the Team Needs to Improve:

While our technical progress and open-source platform help with cost benefits, we recognize that we need to improve our financial responsibility to ensure that the final product remains valuable and cost-effective. Some high-priced electronic components can cause a challenge to budget constraints. An approach to improve this area is to consider more thorough cost-benefit analyses, engage in market research, and explore more resource-efficient designs.

7.2. Four Principles

Below is a table connecting four ethical principles—beneficence, nonmaleficence, respect for autonomy, and justice—to broader context areas. We assume these broader context areas include: **Public health, safety, and welfare; Global, cultural, and social; Environmental** and **Economic** considerations

7.2.1. Four Principles Table

	Beneficence	NonMaleficence	Respect for Autonomy	Justice
Public health, safety, and welfare	The Scanner supports safe imaging for non-invasive applications	Design avoids harmful practices by sourcing reliable components and uses proper documentation	UI allows users to control scans safely	Implementation allows user-friendly accessibility and reliable performance
Global, cultural, and social	Scanner gives access to advanced imaging technology that can benefit 3D scanning industries	Scans will not harm people due to safety standards during building	The MAVinator will use a simple user interface making it usable by people of all different backgrounds	Since it is open-source, it will be made available to several groups of people
Environmental	design promotes sustainability by using durable components to minimize waste	The design will be aesthetically pleasing, kept indoors, quiet, and resources will be used properly to reduce power consumption	The MAVinator will not use up excessively more space than it needs for the scam	Our scanner will be kept indoors and not affect the environment directly, and it also uses common resources to build
Economic	The scanner will speed up workers preforming scans, cutting down on time costs	Cost effective design would not hinder other projects	We are creating this scanner as an open-source attachment for the Voron printer so the user can adapt it to their needs	The scanner will not affect any users or peoples in an unfair manner

7.2.2. Broader Context-Principle Pair

Our design strongly meets the pairing of Economic Respect of Autonomy. Going so far as to make the project Open Source with python as the predominant language in use for our software interface. Conversely we are lacking in Global social and cultural respect for autonomy because we have not yet researched any ways of making our software more accessible to non-english speakers.

7.3 Virtues

7.3.1. Team Virtues

Integrity: Being honest and transparent in communication, test reporting, and documentation. We have consistently provided truthful updates to our advisor and documented both successes and challenges.

Responsibility: Owning our tasks and deadlines, ensuring that everyone completes their work on time and at a high standard. We create shared timelines and check in regularly to ensure accountability.

Collaboration: Supporting one another by sharing knowledge, assisting with complex tasks, and respecting each other's expertise. We hold weekly meetings to discuss progress, solve problems collectively, and ensure no member is left struggling in isolation.

7.3.2. Individual Virtues

Nathan Reff

- Demonstrated Virtue:
    - Through our senior design work, I believe I have demonstrated strong collaboration skills, particularly through the work in the lab sessions with Daniel. Whenever we met to work on the scanner, we would delegate tasks properly. For example, if one of us was focusing on aligning and tensioning the gantry belts, the other would handle preparing the necessary tools and components, ensuring that our workflow remained smooth and efficient. By rotating responsibilities and working cohesively, we were able to make substantial progress this semester.
- Not Yet Demonstrated Virtue:
    - One virtue I feel I didn't have the opportunity to demonstrate effectively was innovation, particularly in terms of creativity. Most of our project this semester was following a guide in building the scanner, which left limited room for open-ended problem-solving.
    - With the design of the user interface, and the more open ended requirements I hope to use innovation more. I'll have more freedom to introduce innovative ideas into the GUI's layout, workflows and data presentation techniques

Luke Post

- Demonstrated Virtue:
    - This year I have done a good job demonstrating the virtue of commitment to quality. A commitment to quality is important to me and this project because without it, our product will not work well and it won't be something that I would be proud to have

created. I do not want to put out a product that I am embarrassed to put my name on.
    - I have demonstrated this virtue throughout the assembly of the PCBs. They need to be carefully and accurately created to ensure a clean output. If we do not have a clean output then our product will not function properly. After testing the PCBs, we do see a clean output due to the quality of work that was done when assembling them.
- Not Yet Demonstrated Virtue:
    - I have not done an amazing job demonstrating the virtue of having respect for nature. Several times I have used more product or materials than I needed because I made a mistake on the first attempt. These products come from natural materials that get wasted. Our final product will also not give back to the environment in any way.
    - What I need to do to demonstrate this virtue is be more considerate of the materials I am using and do things correctly the first time so those materials are not wasted.

Daniel Ripley-Betts

- Demonstrated Virtue:
    - A high degree of social awareness and teamwork has been demonstrated consistently in this project. There has been a great deal of value put on collaboration and efforts for equitable teamwork in all the efforts I have put into this project. This has paid off in spades and I could not have asked for a better group as the effort has been reciprocated.
- Not Yet Demonstrated Virtue:
    - I struggled to demonstrate courage at certain moments when we faced adversity during the second half of this project. It took a very supportive team to keep me motivated when we had to make major changes last minute.

James Peterson

- Demonstrated Virtue:
    - This year I have demonstrated the virtue of listening to feedback. This is true of feedback from teammates, clients, advisors, etc. I

feel I have effectively taken feedback from others to use it to improve upon our project or my understanding of requirements. This has been quite useful in creating the right product for our client.

- Not Yet Demonstrated Virtue:
    - One virtue I believe I can continue to improve on is stepping out of my comfort zone. I have avoided doing things I am not already knowledgeable about and leave that to others you are more experienced while taking on tasks I am already experienced with myself. Improving this virtue could help not only myself to expand my knowledge base but also help the team once I've learned the task.

8. Conclusions

8.1. Summary of Progress

Over the course of this project, we have made substantial progress toward building the MAVinator scanner. Our goals were creating a 3D (XYZ) scanning platform for millimeter-wave imaging, ensuring an imaging volume of at least 300 mm x 300 mm x 300 mm, achieving positional accuracy of 0.5 mm, and developing a user-friendly Python-based interface. Thus far, we have successfully completed the minimum viable product (MVP). Assembling a gantry system adapted from the Voron platform. We also completed initial electronics integration, set up sensor mounting solutions, imaged Marlin firmware onto the MCU. Although we have not fully finished user and systems testing we have a very viable product.

Looking ahead, the best plan of action is to continue the progress we have already made. The MAVinator does not require too much more work to be a highly polished lab tool suitable for any lab environment. Please see 8.3 Next Steps for more details.

8.2. Value Provided

Our client, Dr. Tayeb, has been wanting a new scanner in the CNDE lab as the current scanners are used extremely often, and scans can take long periods of time to complete. Our product also does not need a computer specified just for it to run as we have made it accessible from the web. This will provide more availability for the researchers in the lab to complete scans quickly and efficiently.

Aside from that, we have demonstrated the ability to take a device originally intended for 3D printing, and turn it into a scanning system. Our goal for the project was to make it open source and plug in style so a user can take off the current antenna and replace it with one of their own. All that would be necessary would be to put the code in to control the new antenna. We received the opportunity to do this when we were required to change antennas and DAQs. Now we have both sets of functional code in the project.

Due to our project scope changing in the last two weeks regarding the antenna and DAQ, though, we were not able to test the entire functionality of

the scanner or implement everything we desired. However, we believe from the testing we have done is sufficient to say that our product works. This project has a lot of future work that could be done to make it an even more powerful tool for Dr. Tayeb and the CNDE.

8.3. Next Steps

8.3.1. SAR

Our current implementation of SAR has the minimum number of features required and still contains a couple of bugs. With more time, these are things we would have fixed or implemented to give our user an even better experience.

- Green Heatmap at Layer 0
    - As of right now, layer 0 of the heatmap image is completely green. Some calculation that takes place is causing an error at that layer.
- Depth and Step Size After Calculating SAR
    - Our client has mentioned to us that he would like to be able to change the depth and step size after calculating SAR, not only when the .scan file is uploaded.
- Rendering a 3D Interactable Plot
    - We would like to be able to take all of the slices that can be seen in the 2D heatmap image and turn them into one plot.
    - The user should be able to pan, rotate, and zoom in on specific parts of the plot
- Display the Raw Data on a Plot before SAR Calculations
    - We would like to have the raw data on a heatmap plot as well, then have the raw data replaced by the SAR data once the SAR calculation is run.
    - This would also be necessary for the following next steps to happen.
- Perform Extra Manual Processing on the Raw Data
    - Crop the data to only see a specified x and y range.
    - Filter the data using high pass, low pass, or any other filters that may make the resulting image better.
    - Be able to apply resampling to the raw data to smooth out some of the sharp changes.

- We would like to able to specify what coloring scheme is used on the SAR heatmap instead of ‘jet’ being the only choice.

8.3.2. Modular Classes

- DAQ and Radar
  - Writing abstract base classes for both the Radar chip and DAQ would have helped us this semester to more easily switch from the previous radar to the new one as well as swapping our DAQ.
  - Each implementation of a radar chip would have its own concrete class that implements the interface laid out in the abstract base classes for the DAQ’s and Radar respectively. This would allow each class’s slight nuances to be swapped when switching devices.
  - This would allow for easy swapping of radars and the DAQ.

- Scan Pattern
  - Implementing an abstract base class would give more formal requirements for developers to create their own scan patterns.
  - Implementing a concrete class for each different type of pattern gives the user more flexibility. Some patterns may include fortran-like, circular, and triangular patterns.
  - This would allow for easier switching of scan patterns down the line, potentially from within the UI as well.
  - There may be potential for scans to occur over the z-axis as well.

8.3.3. UI

- The user interface could always stand improvements for our own purposes as well as client desires.
  - Upgrading the javascript libraries we use could enhance the overall feel of the user interface.
  - Implementing more instances of Flask forms may also help with data loss on page refreshes.
  - We would also like to upgrade the positional sliders to move accurately with the printer’s given feedrate, displaying the target end destination of the sensor at the same time as the current position.

9. References

[1] "VORON2.4," *vorondesign.com*. https://vorondesign.com/voron2.4

[2] R. Appleby, D. A. Robertson, and D. Wikner, "Millimeter wave imaging: a historical review," *Proceedings of SPIE*, May 2017, doi: https://doi.org/10.1117/12.2262476.

[3] M. Sabbir, S. Sanjib Sur, S. Nelakuditi, and P. Ramanathan, "MilliCam: Hand-held Millimeter-Wave Imaging." Available: https://cse.sc.edu/~sur/papers/Sabbir_ICCCN20_MilliCam.pdf

[4] M. A. Abou-Khousa, M. S. U. Rahman, K. M. Donnell, and M. T. A. Qaseer, "Detection of Surface Cracks in Metals Using Microwave and Millimeter-Wave Nondestructive Testing Techniques—A Review," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–18, 2023, doi: https://doi.org/10.1109/TIM.2023.3238036.

[5] H. Murakami, T. Fukuda, Hiroshi Otera, H. Kamo, and A. Miyoshi, "Development of a High-Sensitivity Millimeter-Wave Radar Imaging System for Non-Destructive Testing," *Sensors*, vol. 24, no. 15, pp. 4781–4781, Jul. 2024, doi: https://doi.org/10.3390/s24154781.

[6] T.-H. Pham, K.-H. Kim, and I.-P. Hong, "A Study on Millimeter Wave SAR Imaging for Non-Destructive Testing of Rebar in Reinforced Concrete," *Sensors*, vol. 22, no. 20, p. 8030, Oct. 2022, doi: https://doi.org/10.3390/s22208030.

[7] C. Viegas *et al.*, "Active Millimeter-Wave Radiometry for Nondestructive Testing/Evaluation of Composites—Glass Fiber Reinforced Polymer," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 2, pp. 641–650, Feb. 2017, doi: https://doi.org/10.1109/tmtt.2016.2625785.

[8] Bengisu Yalcinkaya, E. Aydin, and A. Kara, "Millimeter-Wave SAR Imaging for Sub-Millimeter Defect Detection with Non-Destructive Testing," *Electronics*, vol. 14, no. 4, pp. 689–689, Feb. 2025, doi: https://doi.org/10.3390/electronics14040689.

[9] “Zaber Technologies,” *Zaber.com*, 2025. https://www.zaber.com/products/xy-xyz-motorized-stages/XYZ (accessed Apr. 19, 2025).

10. Appendices

Appendix 1 - Operational Manual

This section contains descriptions and images of our scanner operations. It will walk through how to operate the scanner and outline key details to inform the user of all of its functionality.

A.1.1. Connecting to the UI

On startup the pi will load the software to run the server. So, all you need to do after the pi is up and running properly is enter the device's IP address into your browser to pull up our web-based UI. The IP address is dependent upon the device and network so you must know this in order to connect to the UI.

A.1.2. Move Page

The Move page is the page that you will be greeted by after connecting to the server. This page is used for moving and homing the scanner head (Figure A.1.2).

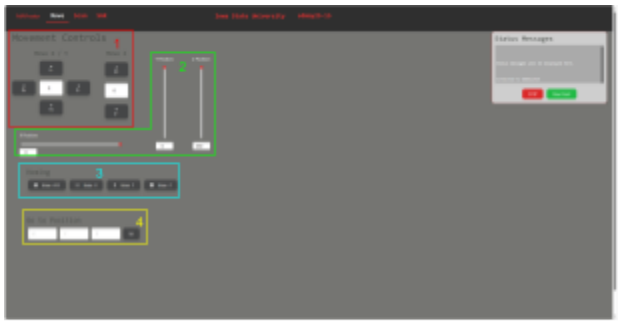


Figure A.1.2-1

Box 1 is used to move the X, Y, and Z axis with a given value. Just enter the value in the box and click the respective button to move the sensor. Box 2 will display the current sensor location dynamically for the user's consistent awareness of where the sensor is. Box 3 is for homing the sensor. There are options to home each axis individually or it can home them all at one. Box 4 is for entering a position to move the sensor to in one motion. Simply enter the coordinates desired and click go and it moves the sensor there.

A.1.3. Scan Page

The scan page is used to configure and perform scans. There are a number of inputs for this page for the user to provide.

Figure A.1.3-1

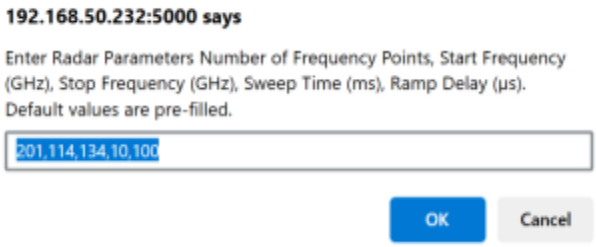
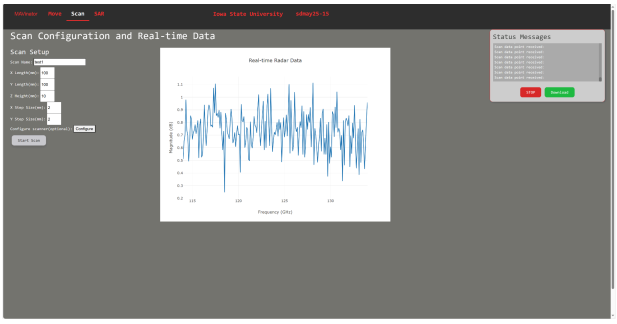


Figure A.1.3-2

Looking at figure A.1.3-1 to begin, there is an input for the scan file name to be saved. Next are the dimensions of the scan, including the X and Y distances to be scanned and the Z height of the object being scanned. Lastly is the step size that the sensor should use during its scan.

There is also the option to configure the scanner (Figure A.1.3-2). There are default values so this configuration is optional. To configure the scanner there are a few parameters to provide. These parameters are the number of frequency points, start frequency, stop frequency, sweep time, and ramp delay.



After clicking the start scan button, real time data from the scanner will be displayed on the plot on the screen. This can be used to ensure the scanner is working as expected and to get a general idea of output from the scanner.

A.1.4. SAR Page

The SAR page is for presenting scans in a human understandable format. It takes a .scan file as input to display a 3D representation of the scan results.

Figure A.1.4-1

Figure A.1.4-1

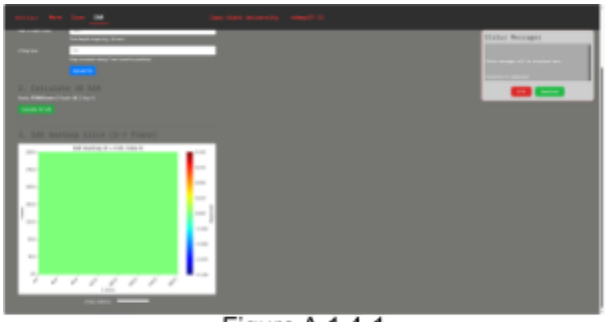


Figure A.1.4-1

The page takes a .scan file as an input, a max z-depth for the SAR to display, as well as z step size for the SAR to display Z depths (Figure A.1.4-1).

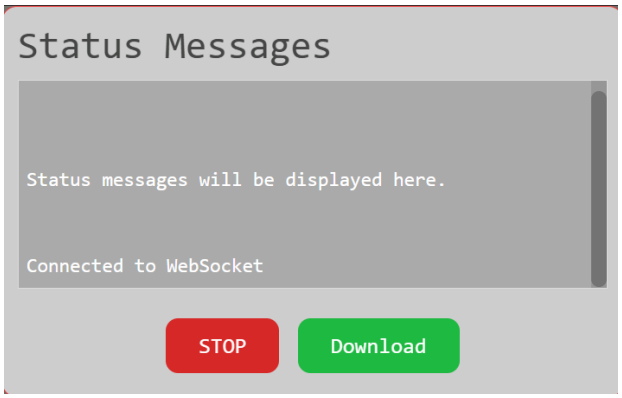
After entering these parameters and clicking “upload file”, a ready indicator will appear above the Calculate SAR button (Figure A.1.4-2). This will allow you to calculate the 3D SAR display.

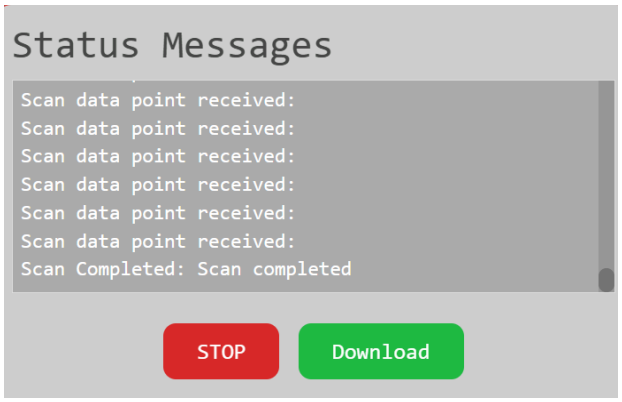
The third section then displays the heatmap of the given .scan file for the user to view and understand (Figure A.1.4-3). The user can use the slider at the bottom to display different Z depts of the scan, moving at the step size given before.

A.1.5. Status Box

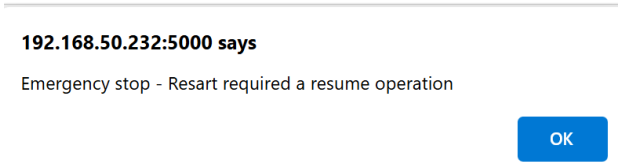
The Status Box is located on every page floating in the top right corner. Here you will see messages displayed regarding the operation of the scanner.

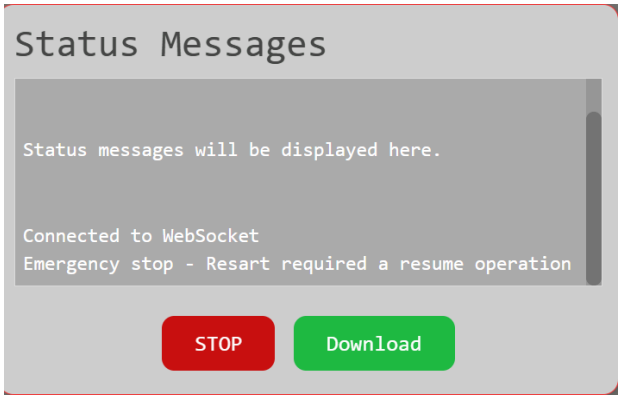
Additionally, this status box is where you can perform an emergency stop and download the most recent .scan file with the respective buttons.





The status box contains a text box to display status messages such as scan progress and two buttons to perform an emergency stop and to download the most recent .scan file.





This will pop up when an emergency stop is issued to inform the user of a required restart. A status message will be displayed in the status box as well.

Appendix 2 - Alternative/Initial Version of Design

This section details the significant design iterations and changes that occurred during the MAVinator project, highlighting the rationale behind each adjustment.

A.2.1. Initial Sensor and Radar Setup

- **Description**: The initial design involved a specific millimeter-wave sensor and associated radar board. This setup was integral to the early stages of testing and development.
- **Change**: Due to unforeseen circumstances, the original radar was required for another project within the CNDE lab. This necessitated a switch to a different radar unit.
- **Rationale**: The need for the original radar elsewhere was beyond the project team's control. The change was unavoidable to ensure the lab's overall needs were met.

A.2.1.2: Prior PCB Layout

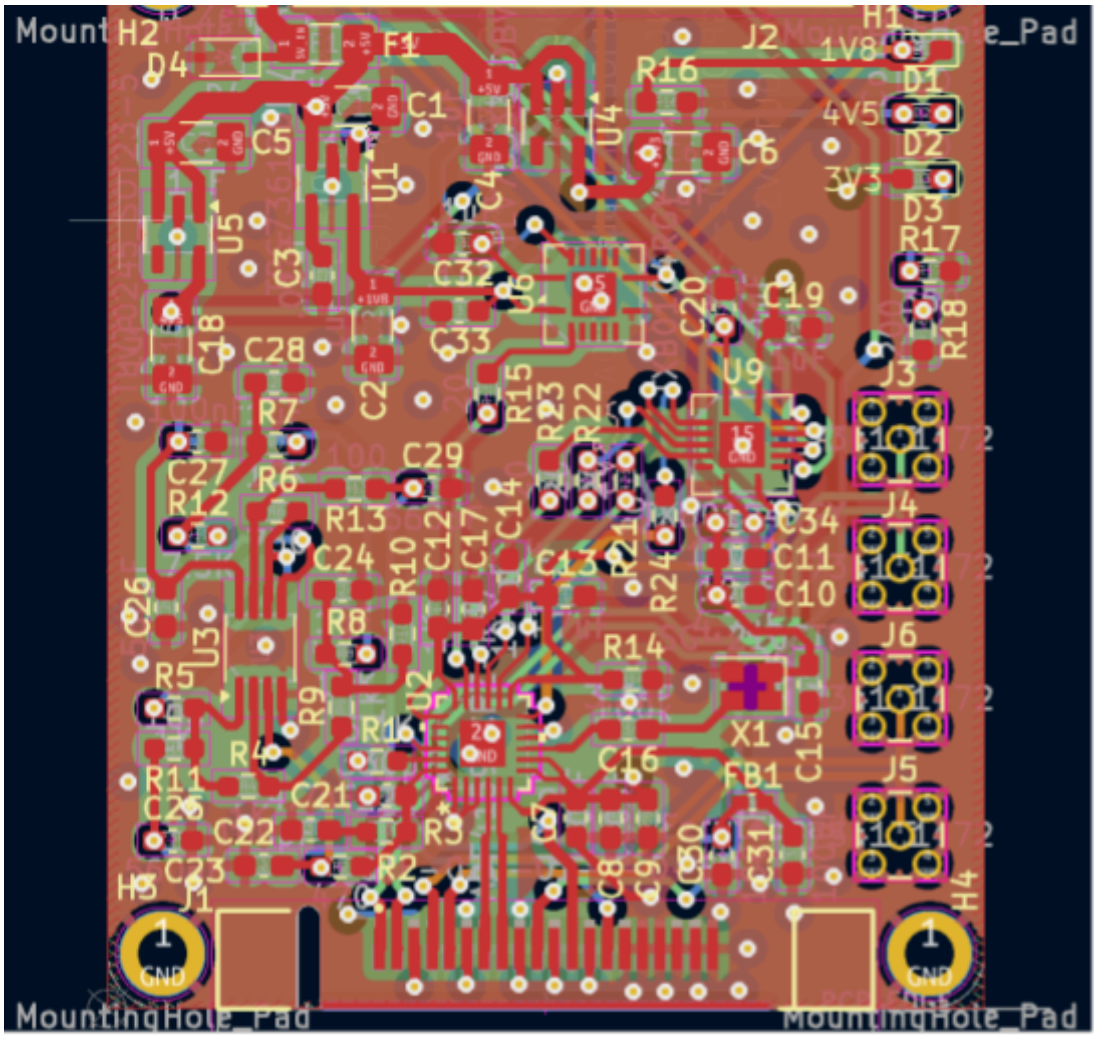


Figure A.2.1-1: Control Board Layout

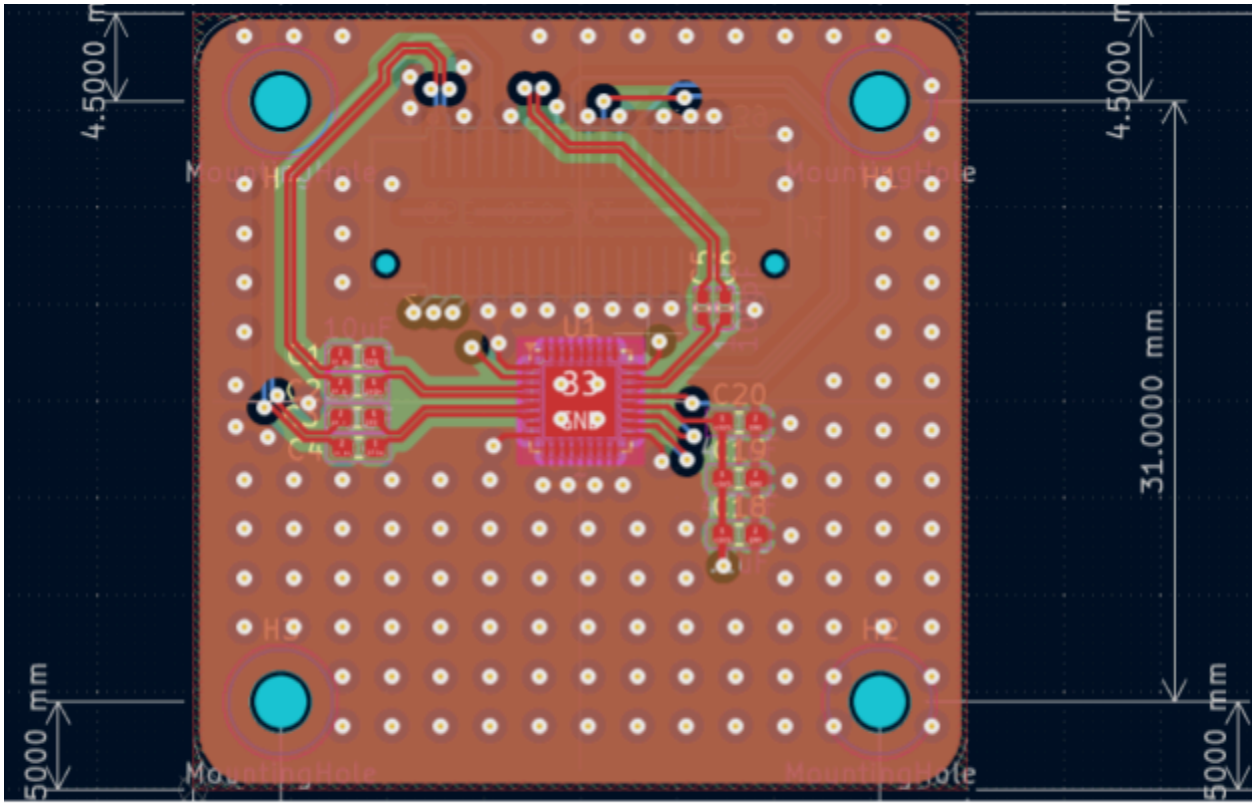
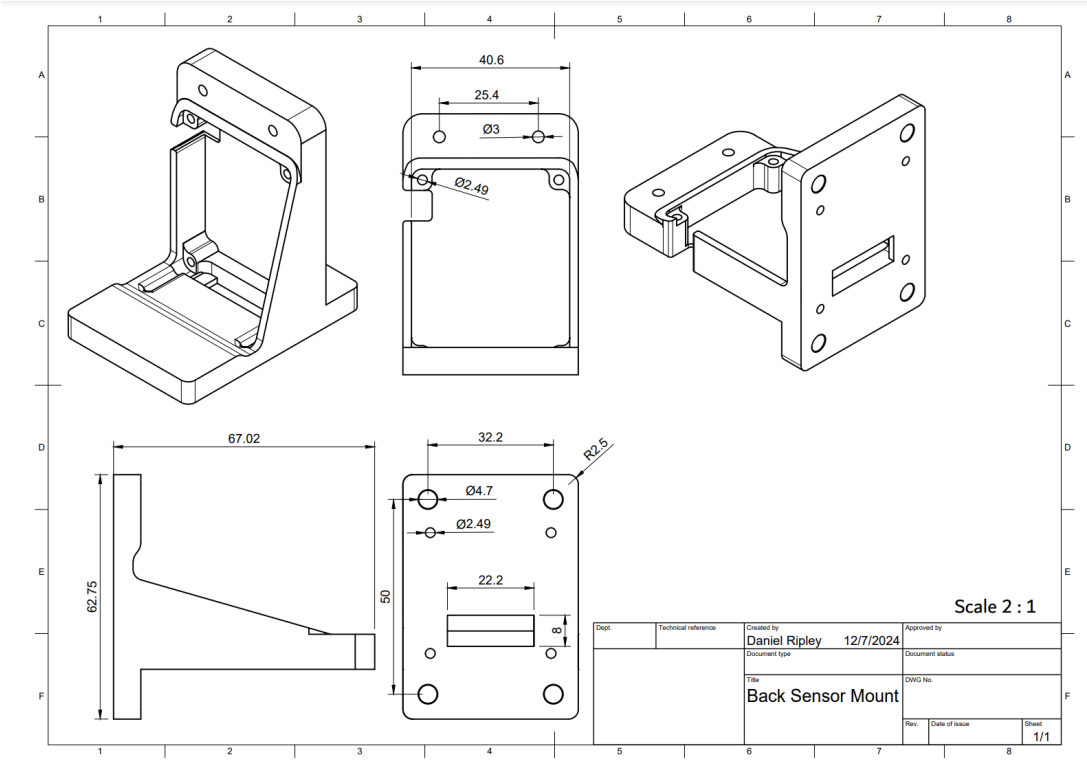


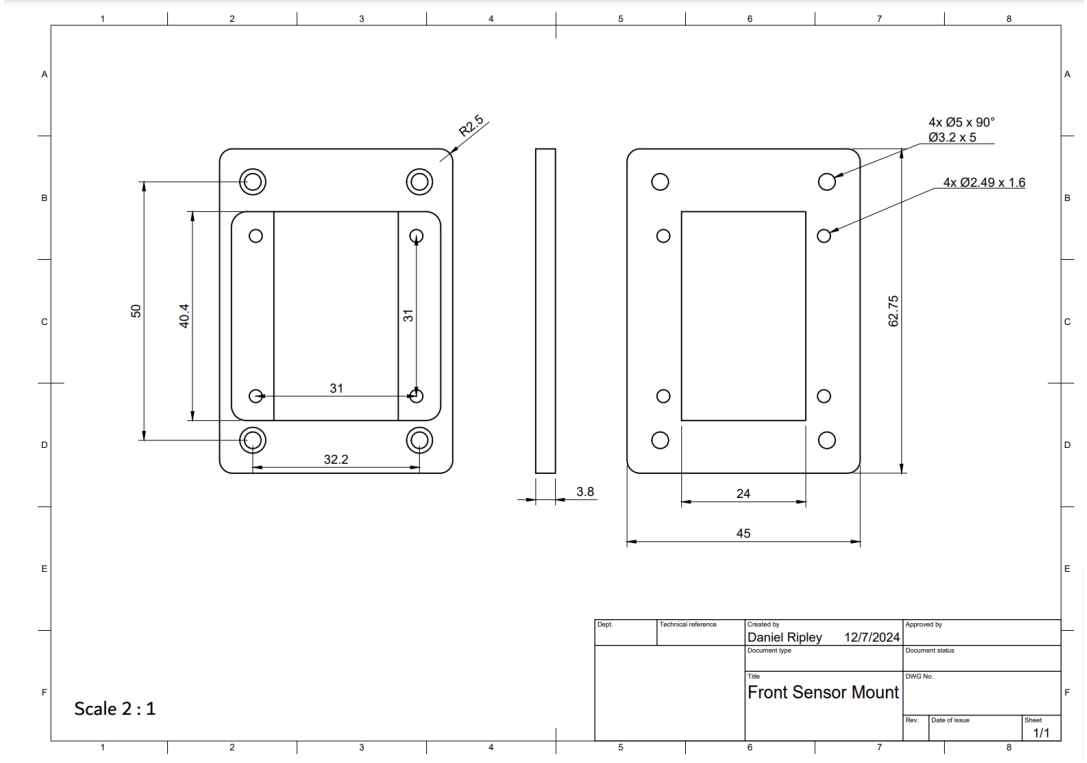
Figure A.2.1-2: Radar Board Layout

A.2.2. Sensor Mount Redesign

- **Description**: An initial sensor housing/mount was designed based on the dimensions and specifications of the original radar unit.
- **Change**: The switch to the new radar required a complete redesign of the sensor mount. The new radar had different dimensions, connection points, and physical requirements.
- **Rationale**: The new mount had to securely hold the new radar, ensure proper alignment for scanning, and fit within the existing constraints of the Voron motion system. This change was a direct consequence of the radar swap.

A.2.2.2: Sensor Mount Drawing





A.2.3. Codebase Adjustments

- **Description**: The initial software development was geared towards interfacing with the original sensor and its data acquisition methods.
- **Change**: With the new radar and DAQ, significant portions of the codebase needed to be rewritten. This included changes to data acquisition, signal processing, and control logic.
- **Rationale**: The new hardware had different communication protocols, data formats, and triggering requirements. The software had to be adapted to these new parameters to ensure proper functionality.

A.2.4. Digital Acquisition (DAQ) Device Swap

- **Description**: The project initially planned to use a National Instruments (NI) DAQ device for data acquisition.
- **Change**: Late in the project, it was discovered that the NI DAQ lacked proper Linux support, which was crucial for the Raspberry Pi-based control system. The team switched to a Digilent Analog Discovery 2 (AD2).
- **Rationale**: The AD2 provided the necessary Linux support and a Python SDK, making it compatible with the project's software architecture. This change was essential for ensuring the system could operate as intended.

A.2.5. Early UI Considerations

- **Description**: The team considered building off of a previous LabView UI, building a UI from scratch, and building a modification for an existing library like Octoprint or Klipper.
- **Change**: The team decided on building a Web based user interface similar to the likes of Klipper using Flask.
- **Rationale**: The LabView UI was not user friendly and needed improvements, building from scratch was a good option but left some unknown variables for the client and unfamiliarity with the current UI. Modifying an existing library like Octoprint or Klipper would impose too many requirements and dependencies on the project.

A.2.6. Impact of Changes

These changes led to increased development time, unexpected challenges, and the need for adaptability. However, they also resulted in a more robust and versatile final product. The switch to the AD2, for example, ensured compatibility and long-term maintainability.

Appendix 3 - Other Considerations

Quote to describe the project: “Sometimes you can’t see the mountain behind the hill you are climbing”

- Daniel Ripley-Betts

Appendix 4 - Code

Github repository: https://git.ece.iastate.edu/sd/sdmay25-15

All of the code that we worked on is under Luke/FlaskMavinator

Appendix 5 - Acknowledgements

The MAVinator project would not be where it is today if it were not for a collective effort from many people. This section is where our team would like to acknowledge some of the direct and indirect contributions of people who helped us along the way, and from work others performed in the past.

Aaron McCarville - Aaron’s invaluable work on millimeter wavelength sensors and its original mounting solution was done before we began.

Trent Moritz - Trent has helped us at almost every step of the way, whether that was access to the build space, 3D printing assistance, testing the sensor, or discussing how the scans will actually happen.

Mat Dvorsky - Mat has helped greatly in the testing, troubleshooting, and in his previous works. Without Mat’s SAR code we would have required substantially more time to develop our SAR implementation. The entirety of the .scan format is also Mat’s creation and our implementation is based on it.

Burk Weber - Burk and Trent both work in and around the lab we were building in at the CNDE. Burk was instrumental in high quality 3D prints and bouncing ideas off of. Good company to be around.

Dr. Tayeb - While Dr. Tayeb is both our advisor and our client and was somewhat pressured to help, our team would still like to acknowledge the lessons that Dr. Tayeb shared with us in the process of building the motion system.

Thank you to the above and many more unmentioned heroes who listened to our speeches and supported us through late nights of work on our project.

Appendix 6 - Team Contract

A.6.1. Team Members

- Nathan Reff
    - Motion System Lead
    - Computer Engineering
- Daniel Ripley-Betts
    - Sensor Mount Lead
    - Computer Engineering
- Luke Post
    - Sensor PCBs Lead
    - Electrical Engineering
- James Peterson
    - Software Design Lead
    - Computer Engineering

A.6.2 Required Skill Sets for your Project

Creating the MAVinator scanner has several necessary skill sets. Without these, the product could face design delays and an inefficient product.

- Electrical Circuit Soldering
    - This is essential to this product because the PCBs needed to be assembled. We received schematics and unpopulated boards for both of the PCBs, but they still needed to be accurately soldered to ensure proper operation. The end stops needed wire

extensions as well which required recreating the wire leads from the end stop to the control board.

- Electrical Circuit Testing
    - The PCBs have several sources of potential error which need to be identified. Proper knowledge on how to test and find errors in circuits is essential to the timeliness of the creation of the MAVinator.
- Mechanical Systems Knowledge
    - The assembly of the Voron scanner required a lot of knowledge about mechanical systems. Although the documentation on the assembly process is detailed, the assembly kit still has several advanced features that need to be properly installed including the belt and gantry system and the electronics and wiring.
- Web Application Development
    - The MAVinator will be controlled through a web application that we will make. Therefore it is crucial that this skill set is covered otherwise the MAVinator will not function properly. The GUI for this web app also has to be aesthetically pleasing according to our client beyond just providing functionality.
- Software Development
    - The code controlling the MAVinator that will be sent through the use of the web application must be developed by our team as well. This will be what sends the gcode commands to the Voron printer which is how the printer moves and is therefore vitally important.
- 3D design/modeling
    - A housing is needed to integrate our PCB system into the Voron scanner. To create this we will use 3D modeling software, then print it out on a 3D printer.

A.6.3. Skill Sets covered by the Team

- Electrical Circuit Soldering
    - Luke + Daniel
- Electrical Circuit Testing
    - Luke
- Mechanical Systems Knowledge
    - All

- Web Application Development
    - Daniel + Nate + James
- Software Development
    - Daniel + Nate + James
- 3D design/modeling
    - Daniel

A.6.4. Project Management Style Adopted by the Team

We employ a hybrid approach combining elements of both Waterfall and Agile methodologies to efficiently manage the MAVinator project.

A.6.5. Initial Project Management Roles

In this project we adopted a democratic or participative leadership style amongst ourselves. In this management style everyone is considered equal, and issues or major changes must be adopted by all group members with equal say in the matter. No one group member controlled the project, instead we all followed a logical flow according to our individual understandings of the project.

A.6.6. Team Contract

Team Members:

1) _James Perterson_____ 2) _Nate Reff_____
3) _Luke Post_____ 4) _Daniel Ripley-Betts_____

Team Procedures

1. Day, time, and location for regular team meetings:
    - ❖ We will meet Friday at the university library from 2:30pm-3:00pm
2. Preferred method of communication updates, reminders, issues, and scheduling:
    - ❖ Communication will occur via Discord: https://discord.gg/jzKjzVqc
3. Decision-making policy:

- ❖ Final decisions will be made with majority rule + Rock Paper Scissors & Tayeb for ties

4. Procedures for record keeping:
    - ❖ We will make use of Google Drive, Git, and Discord

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    - ❖ If you can't make it to a meeting let us know, otherwise, please participate wherever possible.
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    - ❖ We will all share responsibility for all aspects of this project, but if a task is assigned or taken specifically, that individual is responsible for a minimum 51% of that task.
3. Expected level of communication with other team members:
    - ❖ Read and react/respond to messages in the general channel on the Discord server
    - ❖ Use the appropriate channels for your messages on Discord. For example, avoid using general group chat channel for non-general or individual communication.
4. Expected level of commitment to team decisions and tasks:
    - ❖ Please make your opinion known & voice heard wherever possible.

Leadership

1. Leadership roles for each team member (flexible and subject to change):

    Luke: Circuit board testing, Tayeb outreach

    James: Voron & Scanner calibration

Nate: Team Organization

Daniel: Team internal communication & Voron build

2. Strategies for supporting and guiding the work of all team members:
    1. Being gently, honestly, and openly critical
    2. Holding each other accountable
    3. If not meeting standards, a direct example will be given
3. Strategies for recognizing the contributions of all team members:
    1. Active: Reflection & Reports, Verbal expression of gratitude
    2. Passive: Author of documents/code, sending a message, notes

Collaboration and Inclusion

Skills, expertise, and unique perspectives each team member brings to the team:

Luke: Experience Soldering, electrical circuit knowledge, experience with microwave scanners

James: Experience with Python development, GUI development, and sonar & IR scanning/calibration

Nate: Experience in python, and with Arduino platform

Daniel: Experience with 3D printing, coding, soldering, older perspective

Strategies for encouraging and supporting contributions and ideas from all team members:

1. No idea is bad idea (brainstorming channel)
2. Do not hesitate to provide honest Feedback, but try to do so in productive ways

Procedures for identifying and resolving collaboration or inclusion issues:

- ❖ post in general or bring it up during a meeting, preferably with as much specifics as possible
- ❖ Alternatively message any of us

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
   - ❖ Get all hardware assembled, & a plan for software
   - ❖ Have fun working together on a large scale engineering project
   - ❖ Learn a bit about professional design & engineering practices
2. Strategies for planning and assigning individual and team work:
   1. Based off interest & skill sets
   2. Based on current workloads
3. Strategies for keeping on task:
   - ❖ Weekly meetings
   - ❖ Trying to ask productive questions

Consequences for Not Adhering to Team Contract

1. How will the team handle infractions of any of the obligations of this team contract?

   First perform a sanity check with other group mates. If they agree, everyone arranges a group meeting to try to resolve the infractions.

2. What will the team do if the infractions continue?

   The group will seek out our Professors (Fila/Shannon), or in some odd circumstances Dr. Tayeb.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) _____Nathan Reff_____ DATE ____05/04/2025_____
2) _____James Perterson_____DATE ____05/04/2025_____
3) _____Luke Post_____ DATE ____05/04/2025_____
4) _____Daniel Ripley-Betts_____ DATE ____05/04/2025_____