MAVinator: A High-Precision Millimeter-Wave 3D Scanner

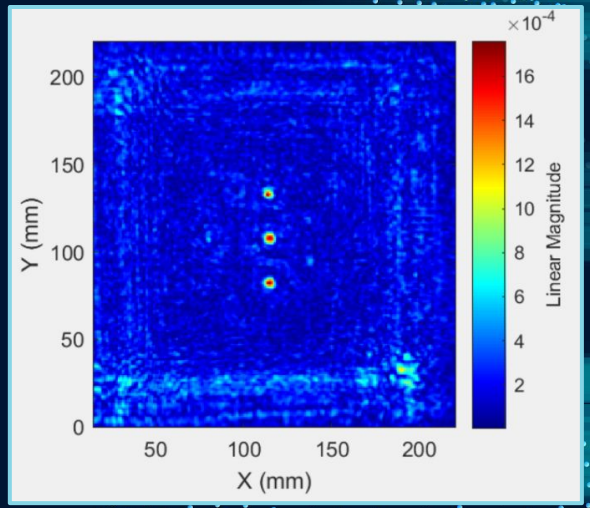
SDMay25-15

Client: Dr. Tayeb | CNDE

Nathan Reff, Luke Post, Daniel Ripley-Betts, James Peterson

Background Information

- CNDE (Center for Non-Destructive Evaluation)
  - Millimeter wavelength imaging
    - Detect defects in metals and other materials
      - To be used in industry or governmental applications
    - Ensuring materials meet safety and reliability standards
- Limited millimeter wavelength scanner capacity
  - Pre-existing scanners are slow
  - Current user interfaces are bulky and hard to update



Technical Requirements

- 300 mm x 300 mm x 300 mm volume
  - positional accuracy of 0.5 mm
  - Operate within 2.2-2.3mm wavelength range

- Develop a Python based user interface.
  - Home and align the scanner
  - Perform automated scans on a uniform cartesian or user-defined grid
  - Configure scanner parameters
  - Perform data collection from a millimeter-wave device
  - Process the data using SAR algorithm and display results

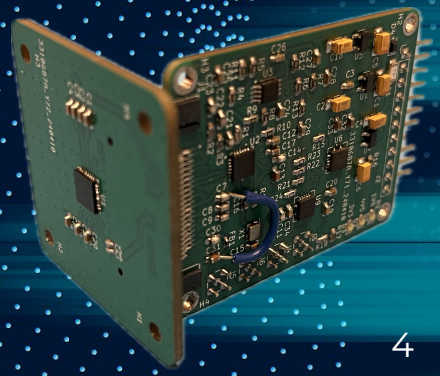
MAVinator Project Overview

Goal:

- Design the physical and digital interfaces for our 3D millimeter wavelength imaging system

Benefits:

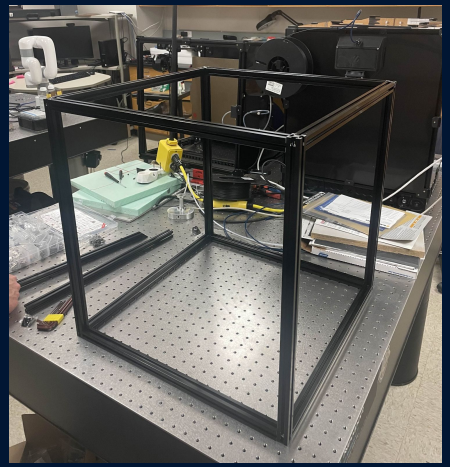
- Compact, affordable, professional, and easy to maintain
- Python based, web-hosted intuitive, user interface

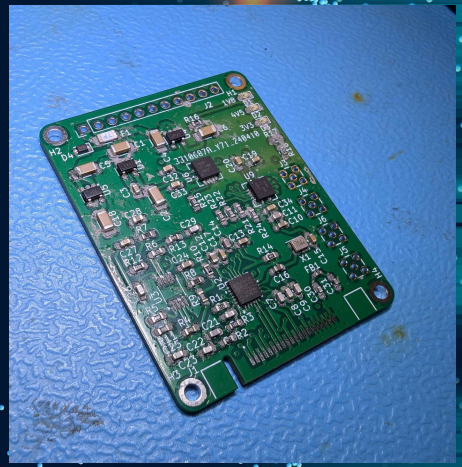


Overview

01	Build + Electronics	➢ Compute/Control Boards ➢ Sensor Housing	➢ Radar + DAQ Setup ➢ Limit Switches
02	Frontend	➢ Html, CSS, JS ➢ Movement	➢ Status Box ➢ Libraries
03	Backend	➢ External Connections ➢ Radar + DAQ Classes	➢ G-Code Pattern ➢ .scan File System ➢ Endpoints
04	Scan, Files, and SAR Implementation	➢ Performing a Scan ➢ Saving Files	➢ SAR Algorithm ➢ Viewing SAR Data

Build





1 — Frame

2 — Belts + Rails

3 — Motors

4 — PCB (Sensor)

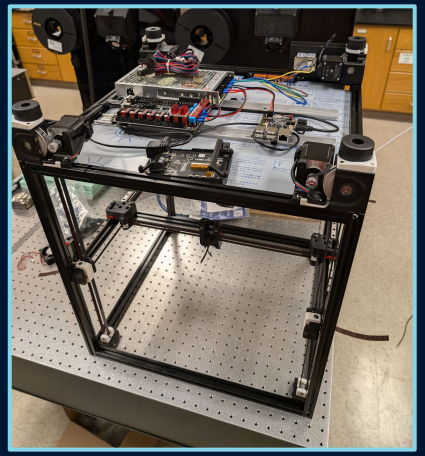
Electronics

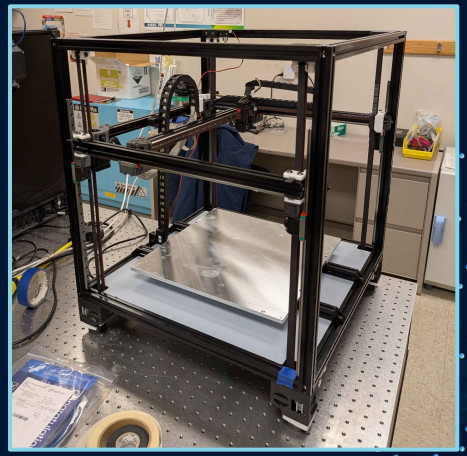
Raspberry Pi	+	BTT Motion Controller	+	Internal Power Supply (PSU)

Automatic movement

➔ Connection from the computer to the raspberry pi and given firmware uploaded to BTT

➔ Successful wiring and utilizing previous User interface for automatic movement



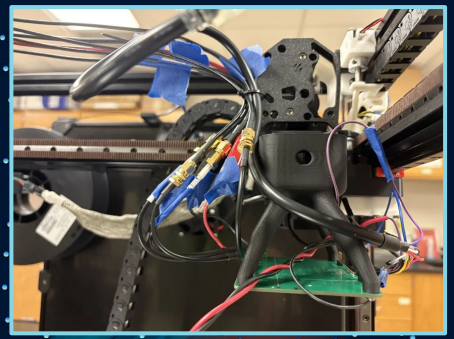


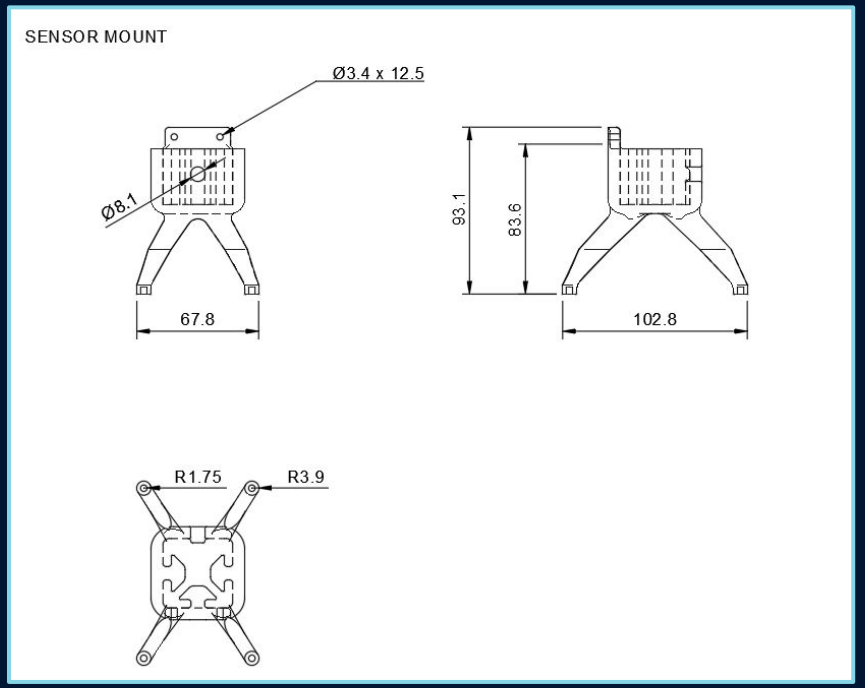
Sensor Housing

Sensor housing was redesigned to support new radar

Current approach: Modify existing model with minimal changes

- increased compatibility with existing equipment
- time constraints





External Connections

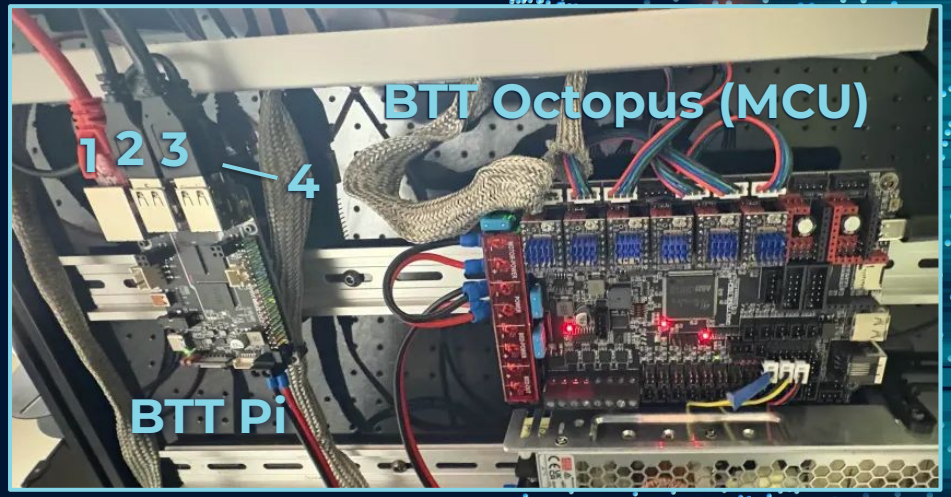
Ethernet (1) - Connecting to the MAVinator and controlling it via web UI.

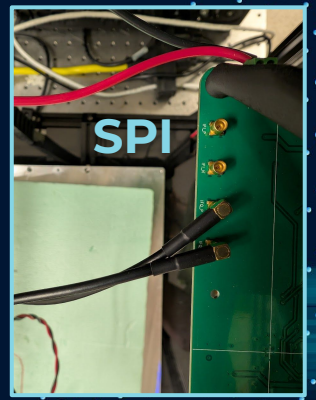
DAQ (2) - Connected to BTT Pi via USB for configuration and data acquisition from SPI

FTDI (3) - Programing radar's registers, triggers DAQ and radar

Serial (4) - G-code to Marlin firmware on MCU

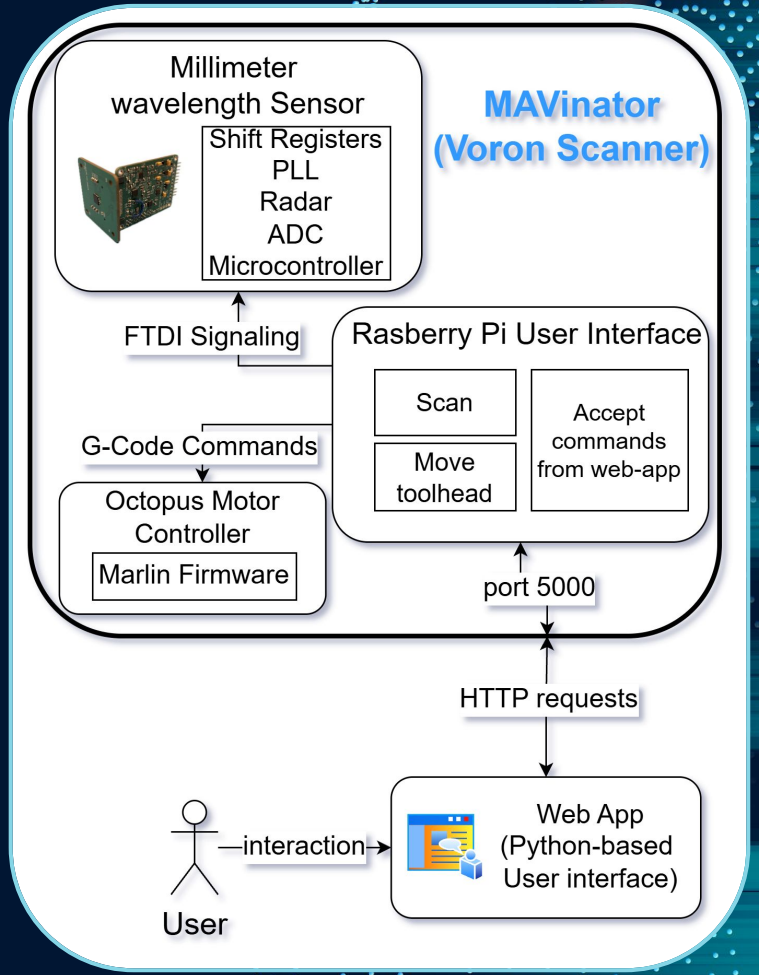
External PSU - Supplies 5V to Radar





Software Overview

- Files
  - HTML, CSS, JS,
  - Python (Flask)
- Connection
  - Frontend to Backend
  - Backend to Scanner
- Pages
  - Move
  - Scan
  - SAR



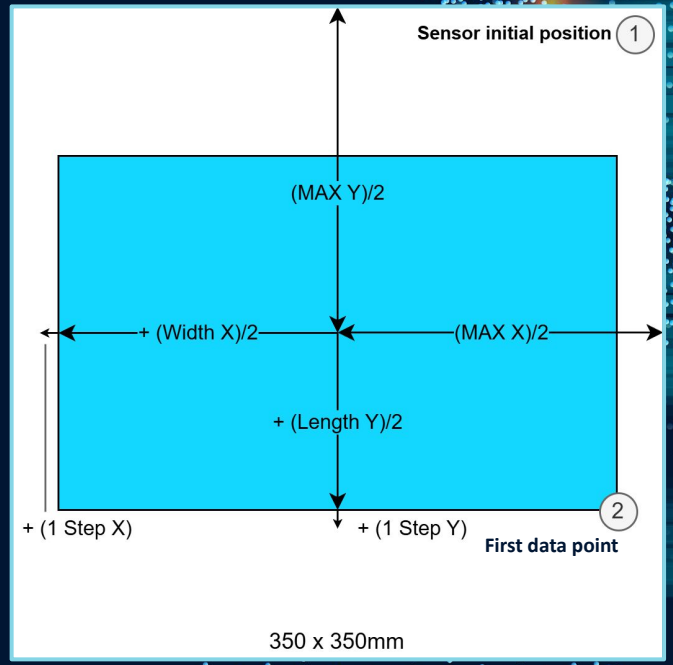
Endpoints

- **Core Role:** The main application file, connecting the web frontend to hardware control, data acquisition, and processing.
- **Key Web Endpoints (HTTP):**
  - **Motion Control:** `/move`, `/goTo`, `/homeOne`, `/homeAll` (POST requests to control sensor position), `/emergencyStop` (POST).
  - **Status:** `/getCoordinates` (GET request to retrieve current position).
  - **Radar Control:** `/initRadar` (POST to initialize radar hardware/simulation).
  - **SAR Processing:** `/SAR` (GET/POST for upload, calculation & display), `/plot/sar_heatmap_slice/<z_index>` (GET for SAR image slices).
  - **Data:** `/download` (GET to download scans).
  - **Pages:** `/move`, `/SAR`, `/plot` (Serve HTML pages).
- **Real-time Communication (SocketIO):**
  - Handles `start_scan` event from frontend.
  - Emits `scan_data_point`, `scan_complete`, `scan_error` during scans.
- **Frameworks:** Built using Flask and Flask-SocketIO.

G-code Scan Pattern Generation

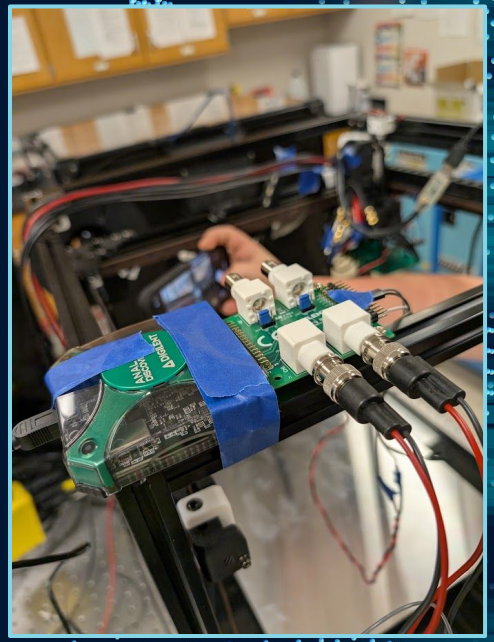
Designed as an independent script for easy modification.

Based off of the idea objects will be centered on build plate and a given Z-height.



Radar and DAQ classes

- **Purpose:** Control custom radar hardware & acquire data.
- **Two Main Modules:**
  - `DigilentDaq.py`: High-level classes (Initialize, Measure, Close, Parameters, Simulation mode).
  - `radarControl.py`: Low-level hardware communication functions (FTDI, Triggering, Data Acquisition).
- **Benefit:** Flexible design supports different DAQ systems & simulation provided you know how to write a class for them.



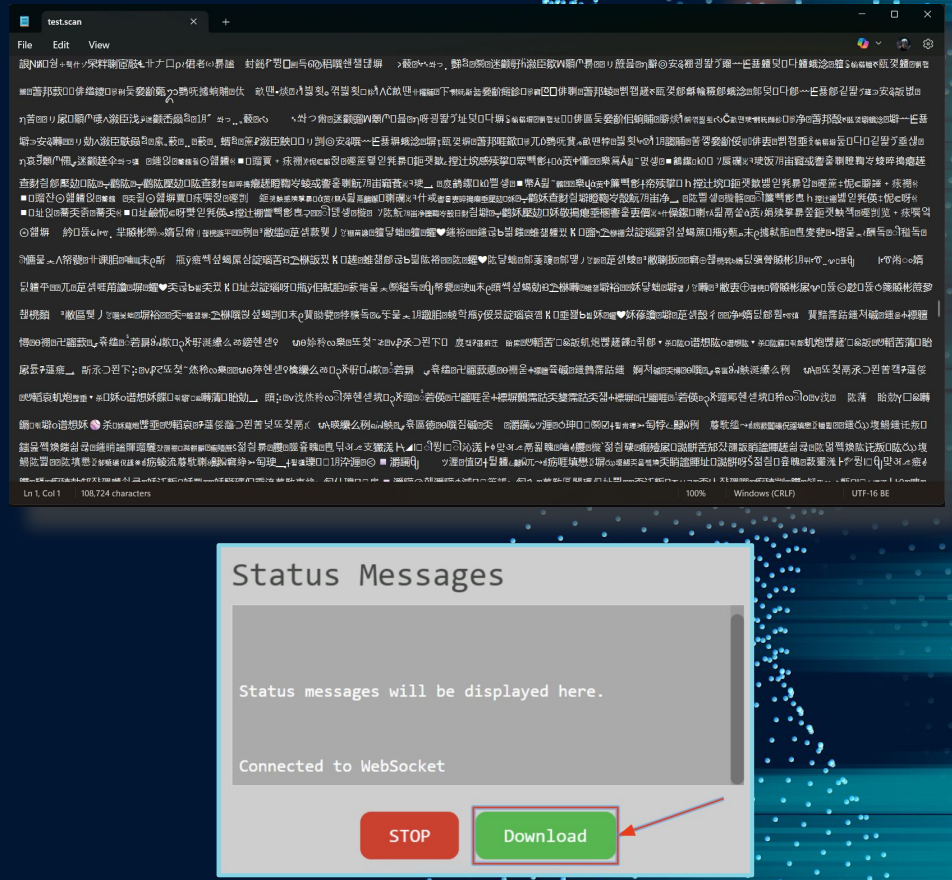
Scan file handling

Proprietary .scan format support

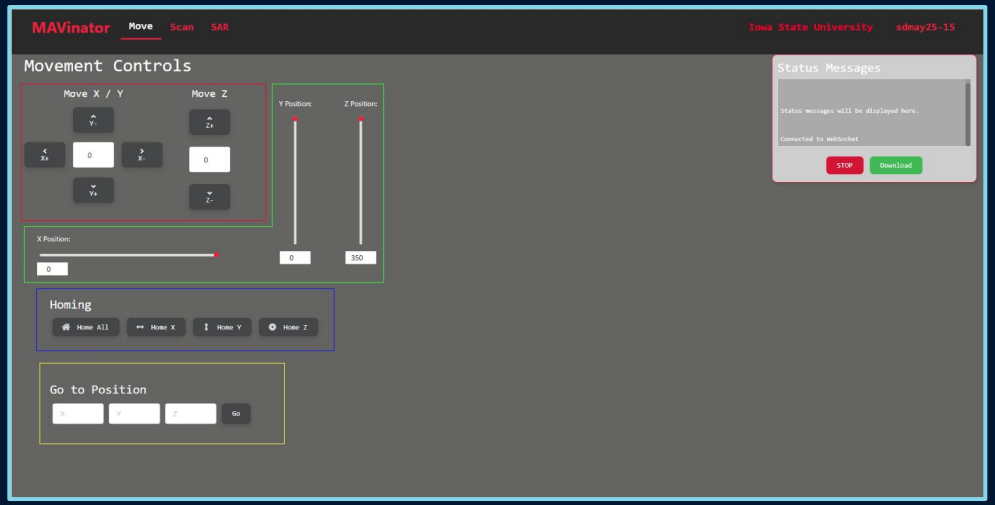
Saved as soon as scan finishes

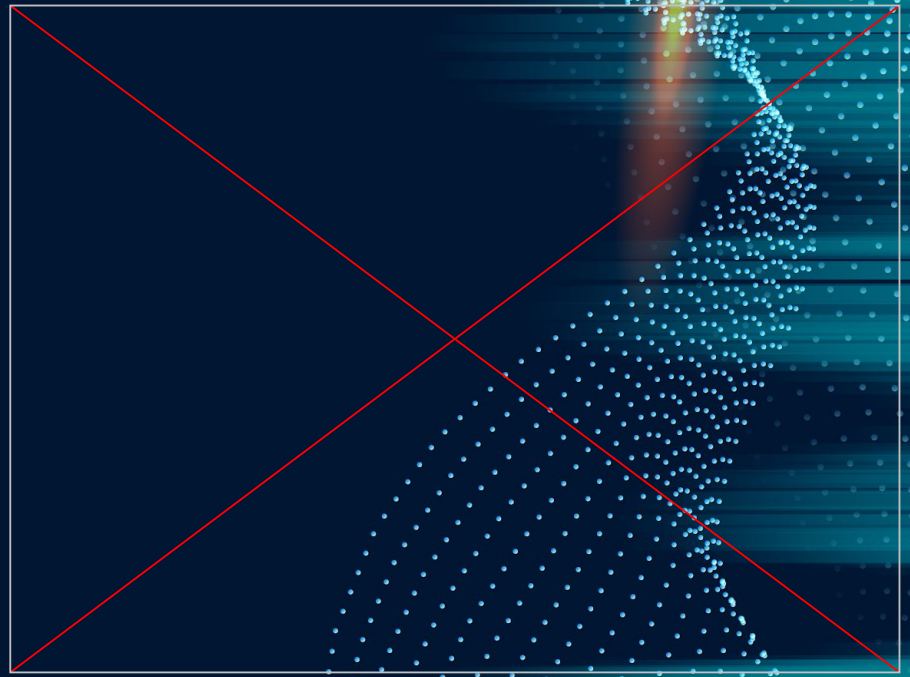
Import and export

Flattens data into point cloud of individual data points

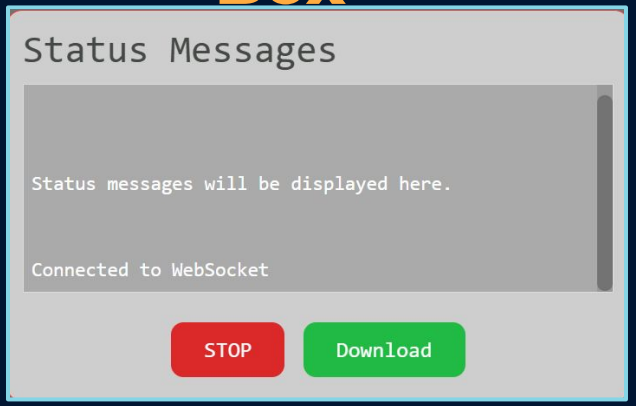


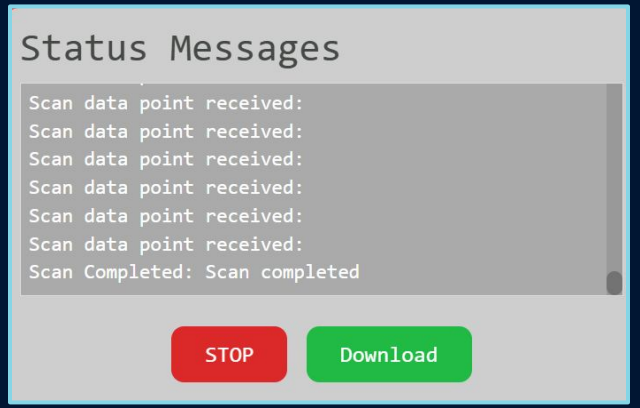
Web Application - Move

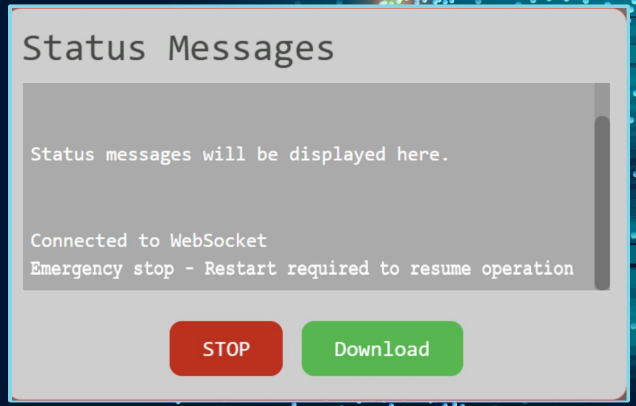




Web Application - Status Box

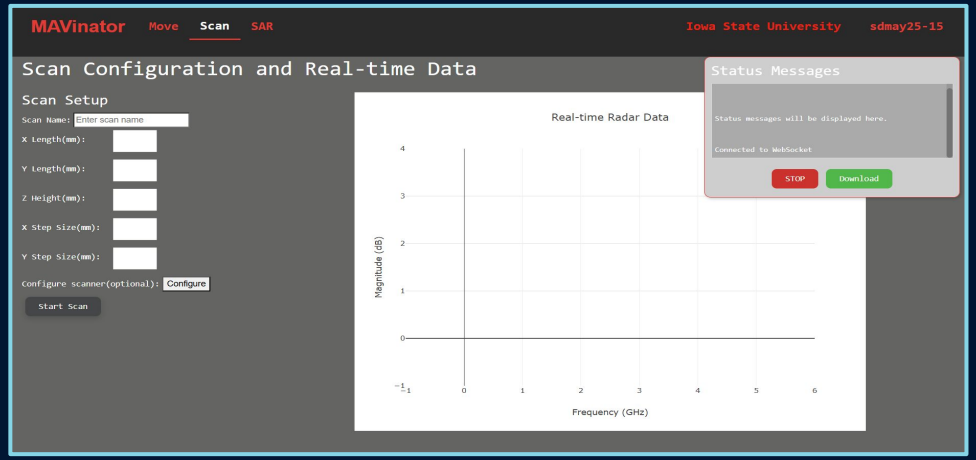


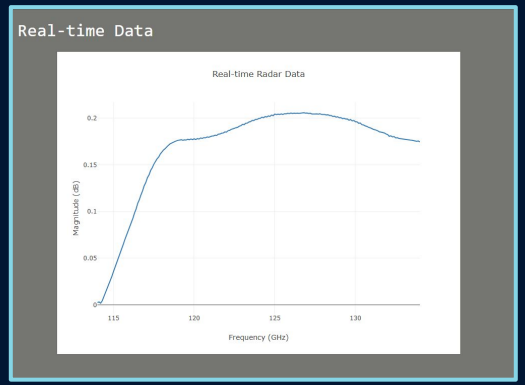


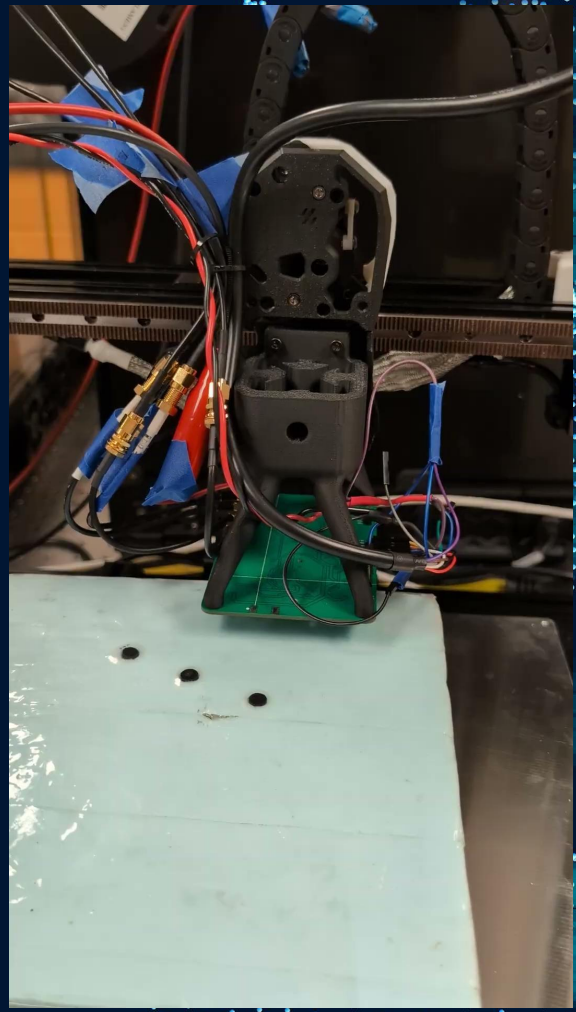


- Uses websockets for real time status updates
- Outputs messages to inform user on scanner operation
- Allows for file saving and emergency stop
- Helps detect errors for debugging
- Persists on all web pages

Web Application - Scan



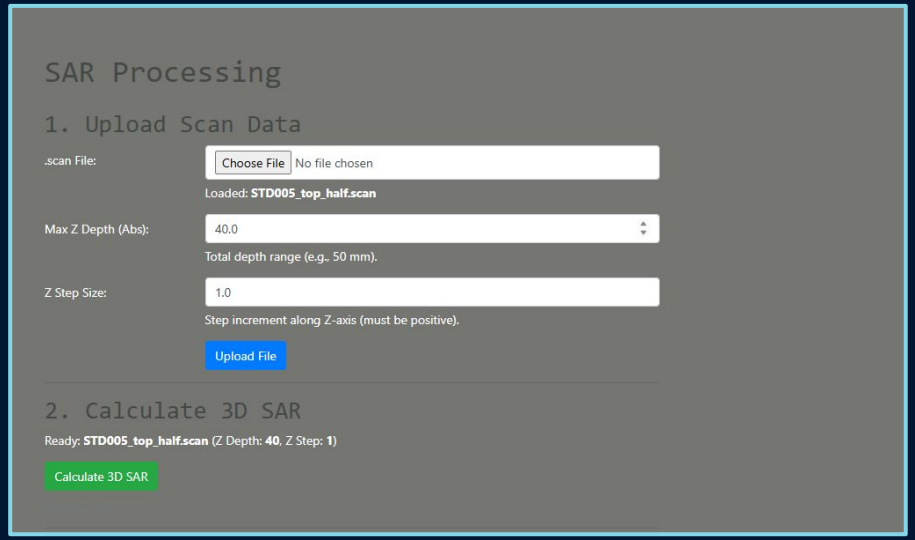


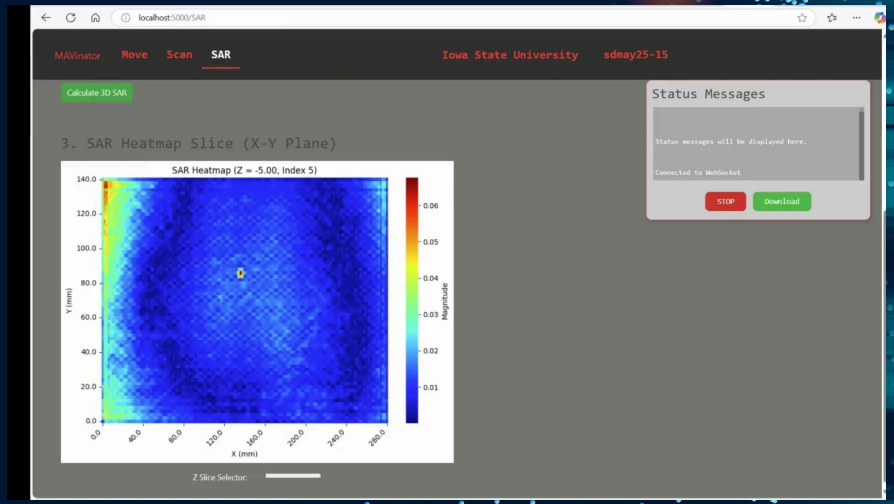


Synthetic Aperture Radar (SAR)

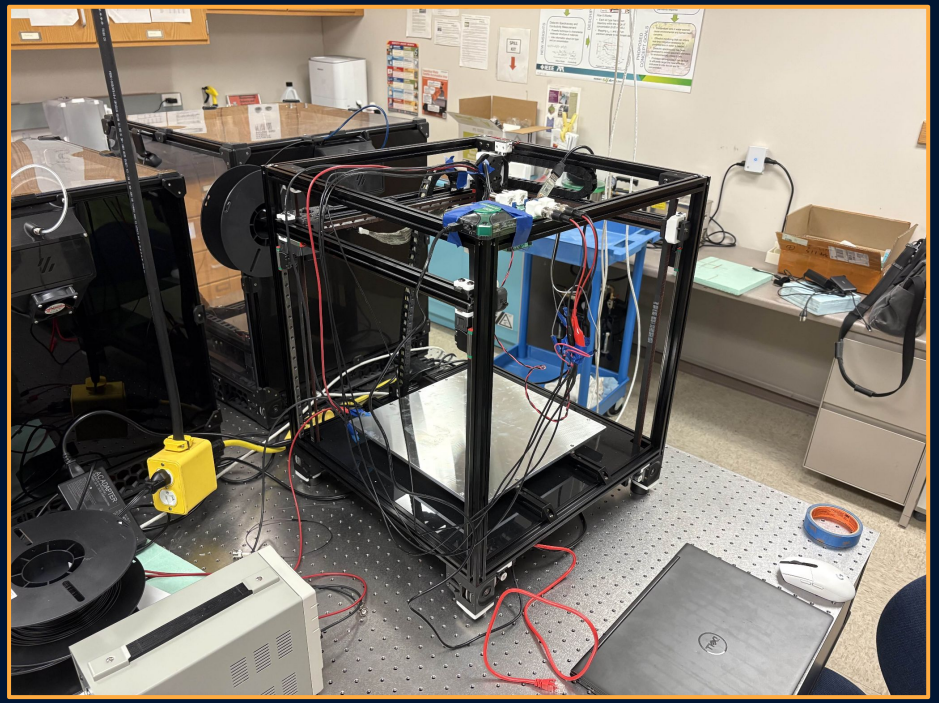
- Angular Resolution is dependent on the wavelength/diameter of the aperture
- Can create an “synthetic aperture” by moving the antenna, taking multiple images, and combining those multiple viewpoints
  - Allows for much better resolution
- Apply processing to turn data into a heatmap

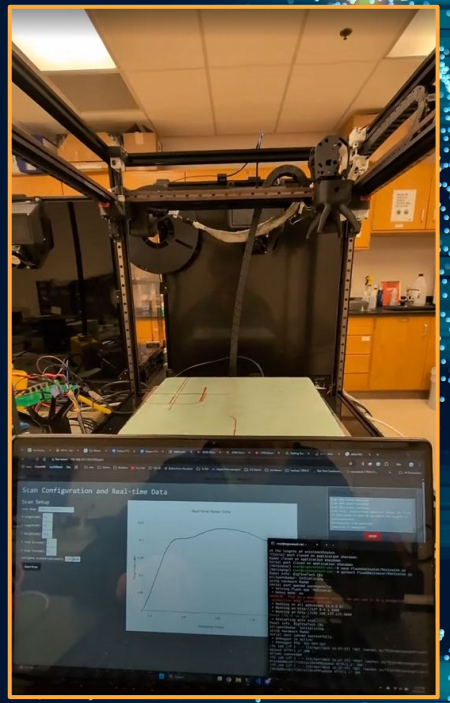
Web Application - SAR

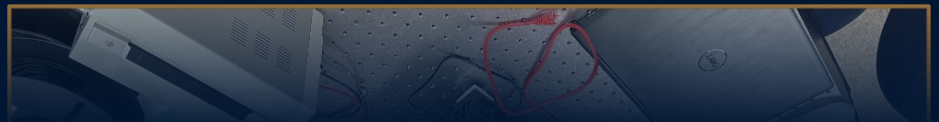


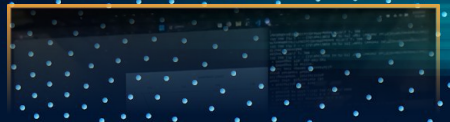


Final Product



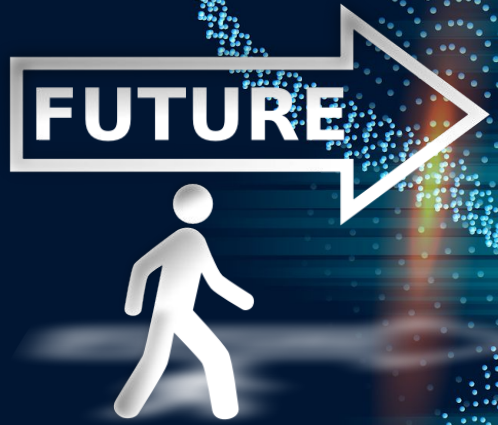






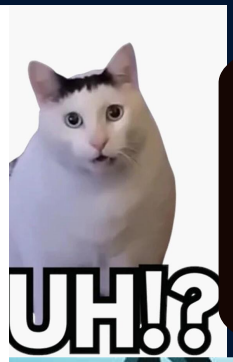
Future Work

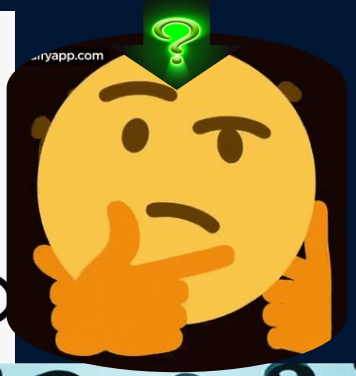
- Modular classes that inherit from base class
  - DAQ
  - Radar
  - Scan pattern
- Further refining the user interface
- Updating file management system
- Adding final physical polish
- Update the BTT Pi access method from IP to URL
- System testing for exporting data collected by comparing to known good scans of same object
- Relative scan pattern generation
- Final testing



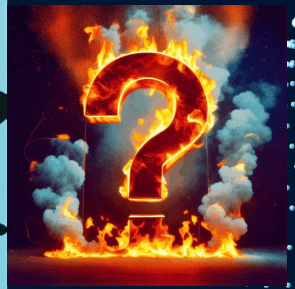
Questions?

Appendix

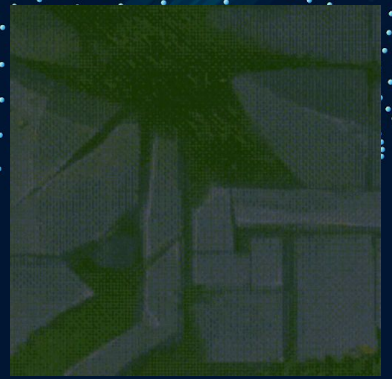

UH!?




QUESTIONS?




??????



First Semester PCB

- Supplied with a PCB and all the components
  - Reflow oven
  - Hand soldering
- Testing
  - Visual
  - Continuity
  - Voltage Level
  - SPI