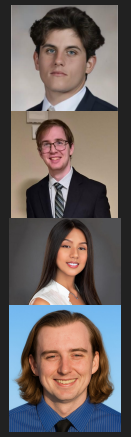
Mini-AWACS Radar Payload Project



Group L14b

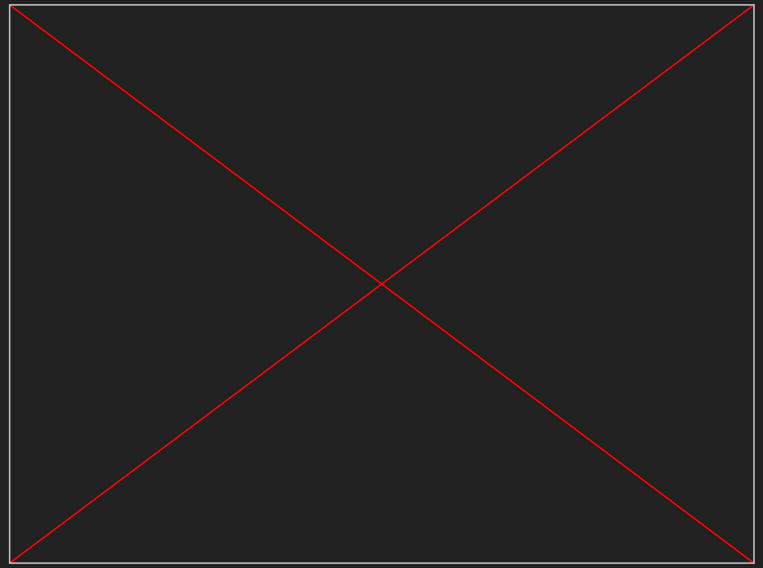
Ido Yaari - Computer Engineering
Phillip Jones - Electrical Engineering
Christina Nguyen - Computer Science
Evan Stevens - Computer Science

Sponsored by U.S. Southern Command

Motivation and Background

- Current drone training methods involve really expensive airborne radar systems, because they need to scan the skies for all airborne vessels. Airborne Warning and Control units or AWACs cost several hundred thousand to operate and even more to make.

- Our sponsor, SOUTHCOM, looks to drastically reduce the cost of the of drone training by utilizing the new Starlink system and building a new system utilizing a marine radar on a smaller plane or balloon going up into the sky.



Goals and Objectives

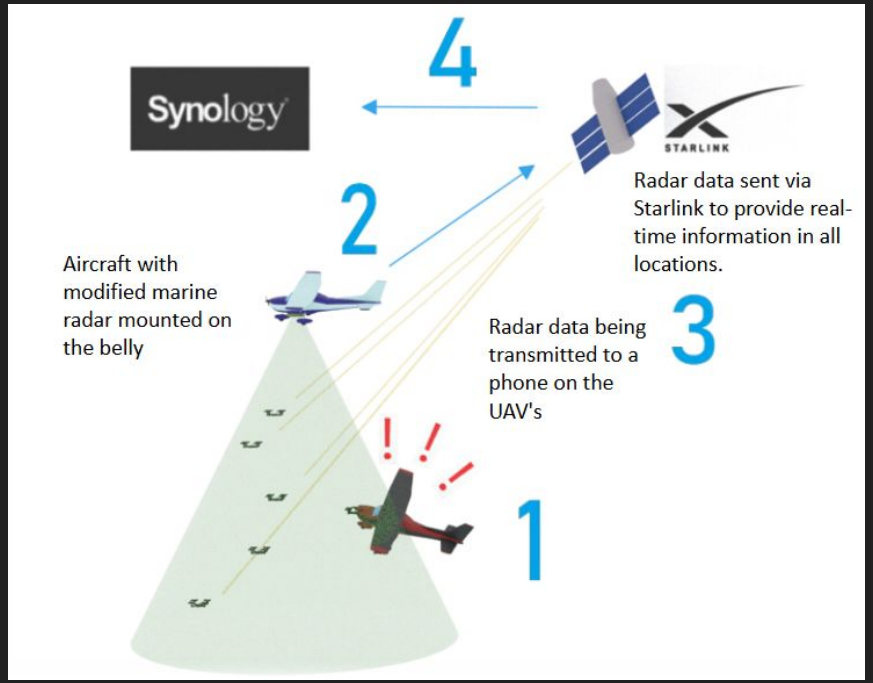
Our project’s objective is to design and build a cost-effective mini-AWACS system that provides small unmanned aerial systems (UAS) with real time radar data for safe operation during training exercises.

Goals:

- Build a motion platform to support and drive the radar.
- Transmit radar and IR data to UAS via Starlink and a cell phone.
- Identify and integrate mechanical systems to achieve precise radar motion.

Original Project System Flow Diagram

- Sending RADAR data via Starlink Mini Terminal
- Receiving RADAR data via Starlink Direct-To-Cell
- To minimize latency, RADAR data will be processed "on-board" into a text format before being sent over Starlink.
- RADAR data will also be periodically sent to our database for users to view



Challenges with Raymarine Radar Integration (Original Hardware Selection)

VCM100 Power Module Problems:

- Inconsistent power delivery to the Raymarine pedestal.
- Required exact wiring conditions and startup voltage to avoid fault state.
- Radar remained in low-power mode even after successful boot from control unit.

Pedestal Boot Failures:

- Radar unit failed to emit or sweep despite power LED indicators being correct.

Closed, Proprietary Protocol:

- Raymarine uses an undocumented binary protocol over RayNet (proprietary Ethernet).
- Unlike Simrad's multicast UDP packets, Raymarine communication required encrypted or opaque handshake sequences.
- Raymarine was not willing to help with project

Worked with Another Group Investigating Raymarine SDK:

- Joint effort to reverse-engineer RayNet communications.
- Their analysis also concluded protocol was non-reversible without official documentation or firmware root access.
- Ultimately chose to pivot to Simrad HALO due to time and documentation constraints.

Large size:

- The Raymarine pedestal was significantly bulkier and heavier than the Simrad HALO20+.
- Difficult to mount securely or transport the system.

Current Project System Flow Diagram

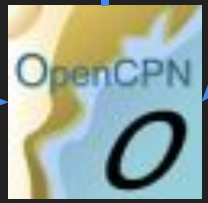
4 Custom Python script to parse the UDP packets being received by the radar.

5 Data packaged in a json packet sent via Starlink to provide real-time information in all locations

Referenced their open source code

1

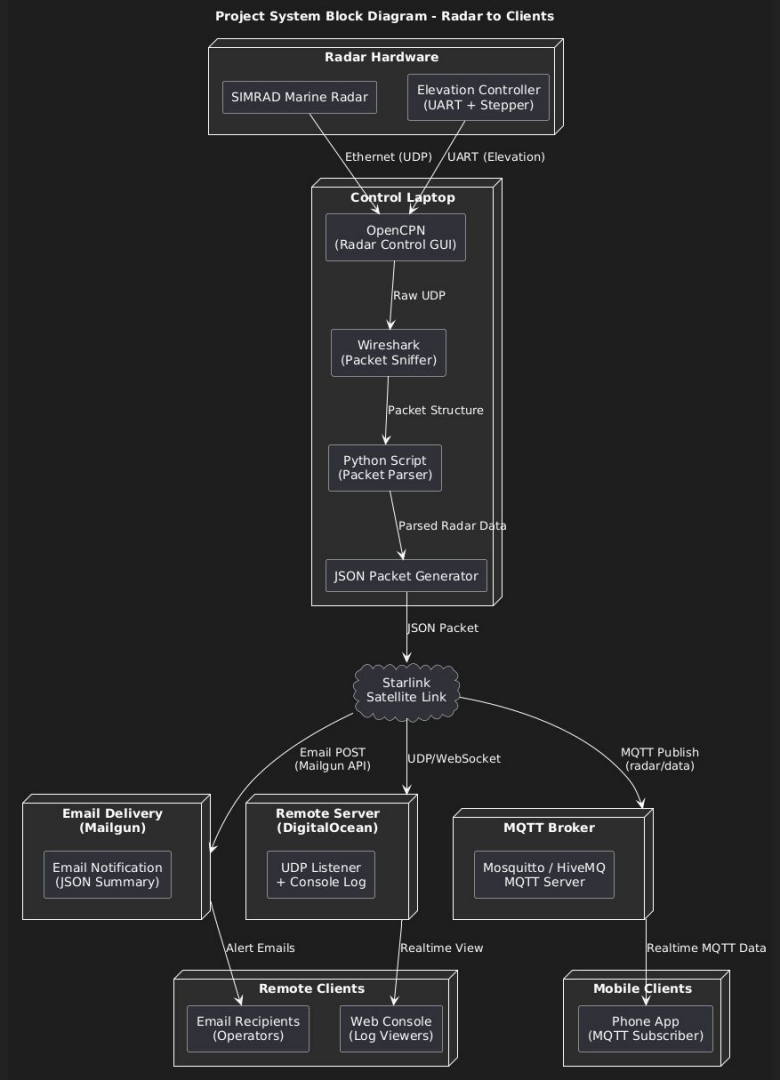
2



3 Used wireshark to acquire the radar's UDP packets for analysis

An unmounted marine radar connected via Ethernet and a mechanical control mechanism using UART, both interfaced with a dedicated laptop.

Utilized open source chartplotter GUI to run radar


Project System Block Diagram - Radar to Clients
Radar Hardware
SIMRAD Marine Radar
Elevation Controller (UART + Stepper)
Ethernet (UDP)
UART (Elevation)
Control Laptop
OpenCPN (Radar Control GUI)
Raw UDP
Wireshark (Packet Sniffer)
Packet Structure
Python Script (Packet Parser)
Parsed Radar Data
JSON Packet Generator
JSON Packet
Starlink Satellite Link
Email POST (Mailgun API)
UDP/WebSocket
MQTT Publish (radar/data)
Email Delivery (Mailgun)
Email Notification (JSON Summary)
Remote Server (DigitalOcean)
UDP Listener + Console Log
MQTT Broker
Mosquitto / HiveMQ MQTT Server
Alert Emails
Realtime View
Realtime MQTT Data
Remote Clients
Email Recipients (Operators)
Web Console (Log Viewers)
Mobile Clients
Phone App (MQTT Subscriber)

Project Specifications

To support remote situational awareness for the radar and simulate the sweeping elevation data, the following needed to be met:

- Capture real-time UDP packets from a SIMRAD Halo20+ marine radar using a dedicated onboard laptop
- Parse radar spoke data into JSON format locally, including azimuth, range, heading, and signal strength
- Move through a 60° top-down arc using a stepper motor controlled via UART
- Record step count and convert to angle in real time; integrate with radar JSON packets
- Send real time captured data via Starlink to a remote DigitalOcean server and Mailgun email endpoint
- Log radar + elevation data to a web console (Socket.IO) and deliver alerts via email (Mailgun API)
- Display parsed radar and elevation data over MQTT to support real-time viewing on mobile and IoT clients

Engineering Specifications Table

Parameter	Specification
Rotating wheel rotation angle	60° range of motion, centered downwards
Motor Rotation Speed	Should complete one 60° cycle in 5 seconds Seconds, at least 12°/s
Starlink connection communication time	Less than 3000 ms
Dimensions	No more than 2 ft x 2 ft x 8 ft

Mechanical Design

There were several different methods we could use to rotate the radar

- Piston/Actuator Wheel Rotation
  - Complex dynamics, with simplified signal control
- Rack and Pinion system to rotate a shell
  - Simplified dynamics, less strain on the motor through the gear ratio
- Directly rotating a shaft with a motor
  - No dynamics, rather steep motor torque requirements with little flexibility

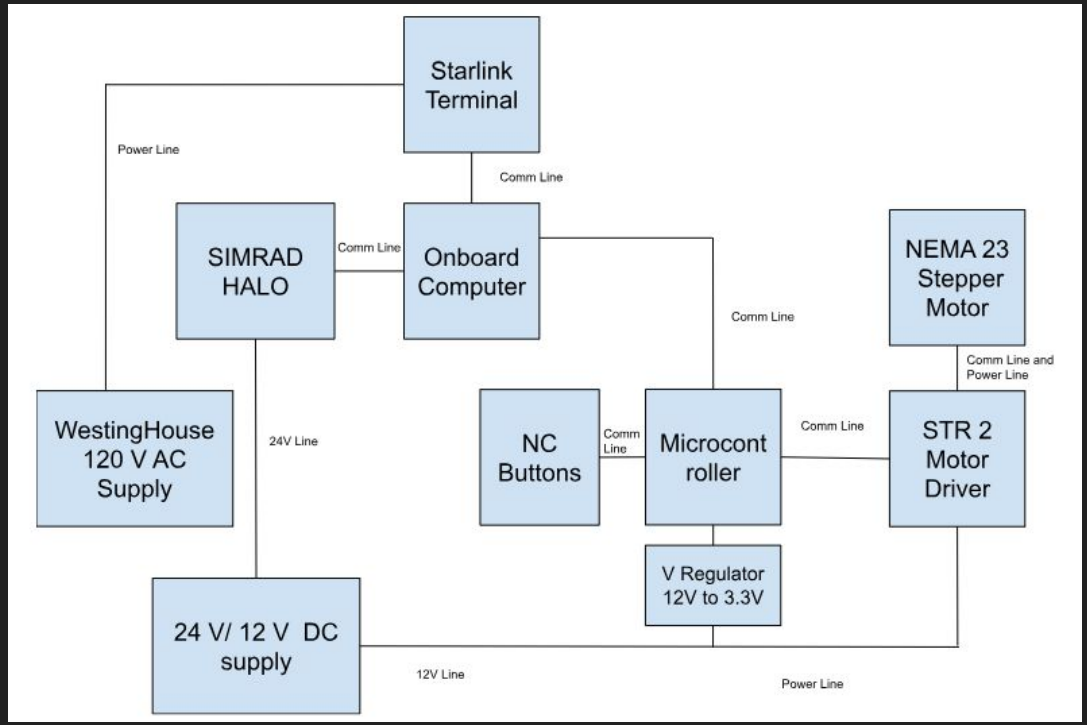
Mechanical Selection

With rack and pinion design, several motor types exist to rotate the gear

- DC motor
  - Simple, high torque, imprecise
- Stepper motor
  - More expensive, high torque, great precision when the motor torque is not exceeded
- Brushless
  - Even more expensive, higher efficiency, low startup torque

Hardware Design

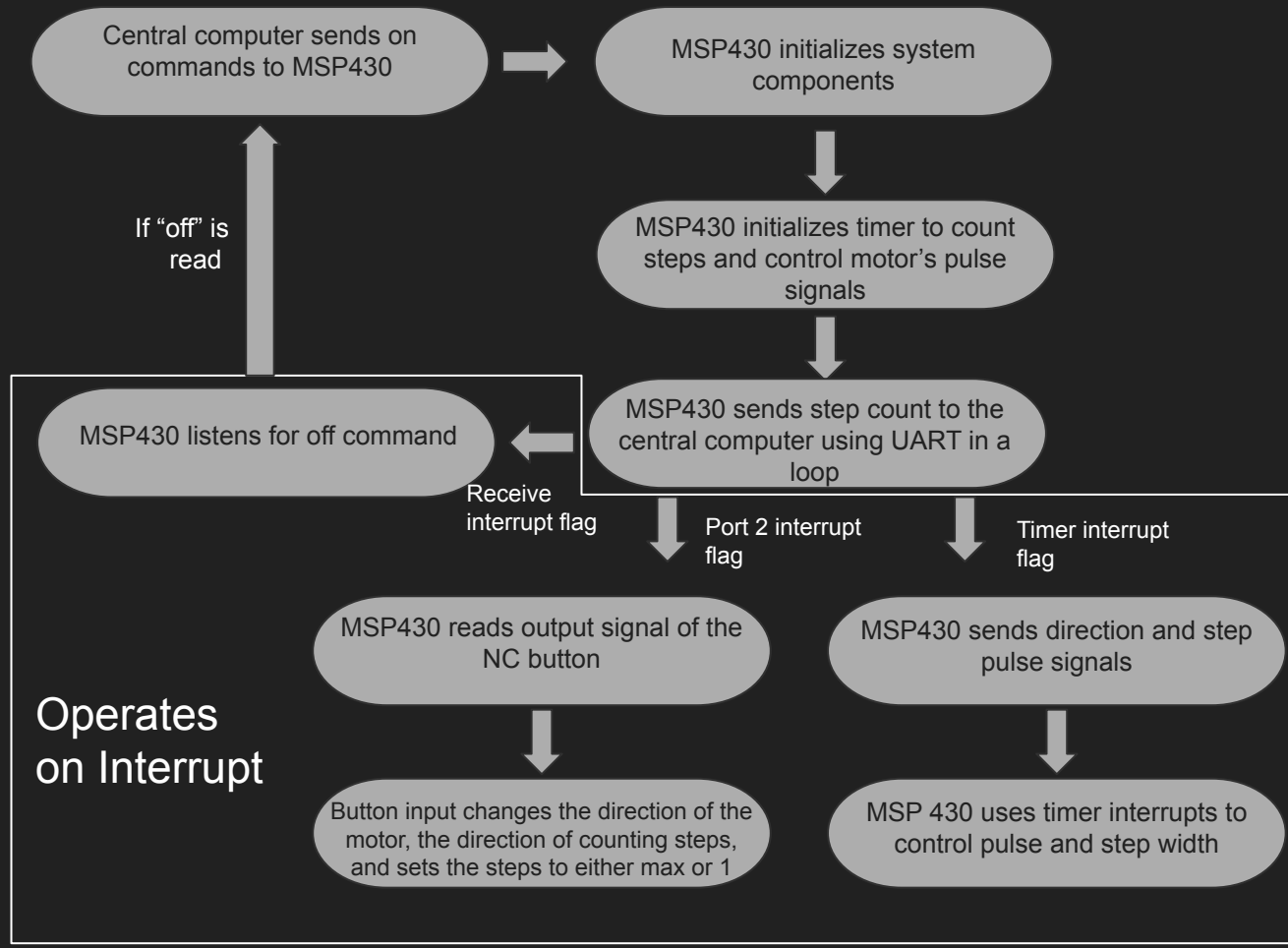
Our General Hardware Layout to fit the project specifications is shown below.



All of the motor and angle communication hardware operates on a 12 V line from our DC supply, the SIMRAD radar utilizes the 24V line from the dual supply from our power output. The starlink is wired independently on our supply provided initially in our sponsor hardware.

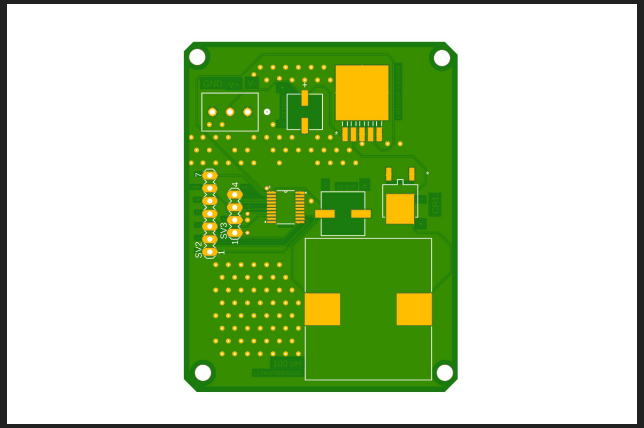
Software Design

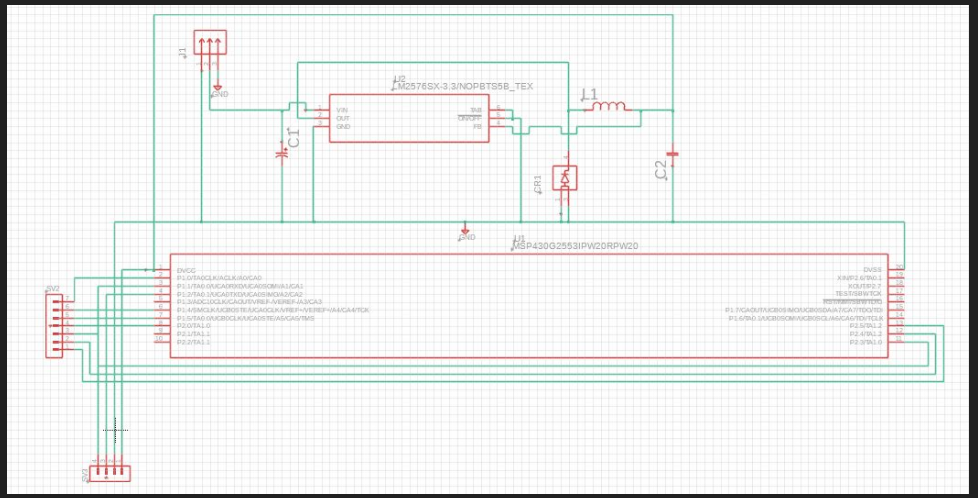
Shown on this slide is our ECE software flow chart. We implemented the following software components to control the project's hardware and send the radar position to the centralized computer to make necessary calculations for radar data.



PCB Design

Although our sponsored project does not have significant PCB Design, our general PCB Layout is shown below (initial PCB design on the left, Schematic on the right). Our PCB provides a method to power the microcontroller communicating with both the motor and the buttons, with 3- pins reserved for the motor and 2 reserved for the encoder, and 2 open pins to implement any other necessary peripherals. We also wired it to enable UART communication on the TXD and RXD pins.





Hardware Selection

Simrad Halo Dome Radar (Provided by UCF)

-Chosen for its compatibility with the free open source chart plotter, OpenCPN with radarpi plugin, which saved us $7,000 from buying the Multi-function display chartplotter

- Chosen for it’s smaller more portable size compared to Raymarine Cyclone

Starlink Roam Mini Terminal (provided by sponsor)

- Enables real-time radar data transfer in remote locations

Samsung Galaxy S24 (provided by sponsor)

- Acts as primary mobile radar interface
- Supports 5G, real-time alerts, and radar visualization

Hardware Selection - Continued

Intel NUC (provided by sponsor)
- Handles local radar data filtering

HP Laptop (provided by sponsor)
- Used by CS team for development and debugging

Westinghouse iGen160s (provided by sponsor)
- Multi hour battery life
- Powers Starlink Terminal

Talentcell 24V/12V Lithium ion Rechargeable Battery
- Light weight (1.39lbs)
- Multi hour battery life
- Powers Simrad Halo and STR2 Motor Driver

DIYhz Normal Closed Button Switch
- Calls interrupt when button is pushed by sweeping rack

Hardware Selection - Stepper Motor

Motor Model Name	Holding torque	Frame Size	Cost
HT08-230D NEMA 8 High Torque Stepper Motor w/ double shaft	2.69 oz-in	NEMA 08	$53.00
HT11-020 NEMA 11 High Torque Stepper Motor	7.08 oz-in	NEMA 11	$58.00
HT17-268 NEMA 17 High Torque Stepper Motor	32.1 oz-in	NEMA 17	$48.00
HT23-601D NEMA 23 High Torque Stepper Motor w/ Double Shaft	269 oz-in	NEMA 23	$110.00

Hardware Selection - Microcontroller

Microcontroller	Pros	Cons
STM32	- Performs well in timing critical conditions. - Power Efficient	- Complex IDE - Tough to learn
Teensy 4.1	- Very Fast - Power Efficient	- More expensive - Limited in its logic
Arduino Due	- Simple to use	- Slower - Not as accurate on timing critical applications
MSP430	- Extremely low power consumption. - Team has familiarity with the MSP430	- Slightly slower

Software Selection - Communication Protocol

Communication Protocol	Pros	Cons
UART	- Ease of implementation - Asynchronous, the most reliable way to run two connected devices on different clocks	- No clock line so timing can be an issue if we add more peripherals connected serially - Slower when passing large data at high speeds
I2C	- Built in clock provides more timing accuracy	- Slower when passing large data at high speeds
SPI	- Fast - Good for high frequency data	- More complex setup - More wiring (space issue)

Software Stack & Tools Used

- **Code Composer Studio** - Used to program and debug the MSP430 microcontroller for radar movement
- **Visual Studio Code**
- **Wireshark** – Captured raw UDP packets from HALO20+ radar over Ethernet.
- **Python (socket, struct)** – Custom scripts to receive and decode radar spoke data. Transformed range/azimuth/elevation data into structured messages.

  Serial Communication (pySerial) – Received elevation angle via UART from stepper motor driver.
- **OpenCPN Plugin (radar_pi)** – Reverse-engineered `NavicoReceive.cpp` to understand range and Doppler format.
- **Node.js + Express** – Backend server handling radar data and WebSocket broadcasting.
- **Socket.IO** – Real-time communication between radar server and client dashboard.
- **PM2** – Process manager for running Node server on remote cloud (DigitalOcean).
- **DigitalOcean** – Hosted our radar processing backend with public IP.
- **Starlink** – Enabled remote field data uplink for testing.
- **Mailgun API** – Used for email alerts and debug output from radar server.
- **MQTT** - used to send real-time data to lightweight clients, particularly to the mobile phone interface for field use.

Radar Spoke Structure & Intensity Values

What is a Spoke?

- A **spoke** is a single directional beam of radar data, representing returns along one **azimuth angle** (e.g., 0°, 0.35°, 90°).
- The radar sweep is made of **~1024–2048 spokes per rotation**.

Spoke Structure

- **Header Info**: azimuth angle, heading, time, range setting.
- **Payload**: array of **range bins**, each bin shows return signal **intensity (0–255)** at a specific distance.

Example:

No detection at binindex[0] == 0 and strong detection binindex[1] == 255

Further Development

Finalize radar casing with updated design

- Finalize plan to mount the Simrad HALO in an aerodynamic case on the bottom of the aircraft

Test the prototype on a small aircraft.

- Air test the system to ensure complete functionality

Bill Of Materials

Component	Name	Cost
Second Radar	HALO24 Radar	$3,099
First Radar	6’ Raymarine Cyclone	$8249.99
Radar Software	Raymarine Software Dev Kit	$500
Data Transmission	Starlink Mini Terminal	$450
Laptop	HP Core i7	$600
Computer	Intel NUC	$350
Portable Power Supply	Westinghouse Portable Power Supply	$130
NEMA Stepper Motor	HT23-601D NEMA 23 High Torque Stepper Motor	$110
Motor Driver	STR 2 Motor Driver	$123
Microcontroller	MSP430G2553	$5

Bill Of Materials

Component	Name	Cost
Portable Power Supply	Talentcell DC Battery	$60.99
NC Buttons	DIYhz NC Buttons	$9.99
Wheel rollers	Steelex 2-inch wheels	$12.99
PCB	JLCPCB	$44
Board Components	Mouser	$100.53
Frame	Wood Frame Materials	$15
Frame	Wood Screws	$10

Work Distribution

Hardware Design - Phillip Jones

Controller & Communication Software Design - Phillip Jones

Mechanical Design - Primary: Ido Yaari Secondary: Phillip Jones

Transformation Algorithm & Software Design - Computer Science Team, Evan Stevens & Christina Nguyen

Mobile phone transmission - Computer Science Team, Evan Stevens & Christina Nguyen

Prototyping & Testing - Phillip Jones & Ido Yaari