# Ultrasonic Object Detector

Baoshan Liang
Brock Dykhuis
Nate Clarke
Nicholas Jacobs
Jonathon Madden

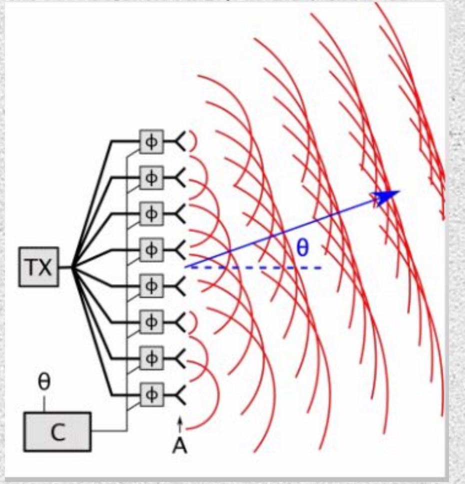
sdmay25-36

Client: Professor Song

# Design

# Project Need and Goal

- **Design an Object Detector to which utilizes a series of Ultrasonic Pulses**
- **Utilize a phased array system to steer for direction scanning**
    - Creating constructive and destructive interference with phase delays
- **Detect reflected Ultrasonic waves, and determine time delay**
- **Calculate distance using time delay**
- **Detect and distinguish between multiple small objects**

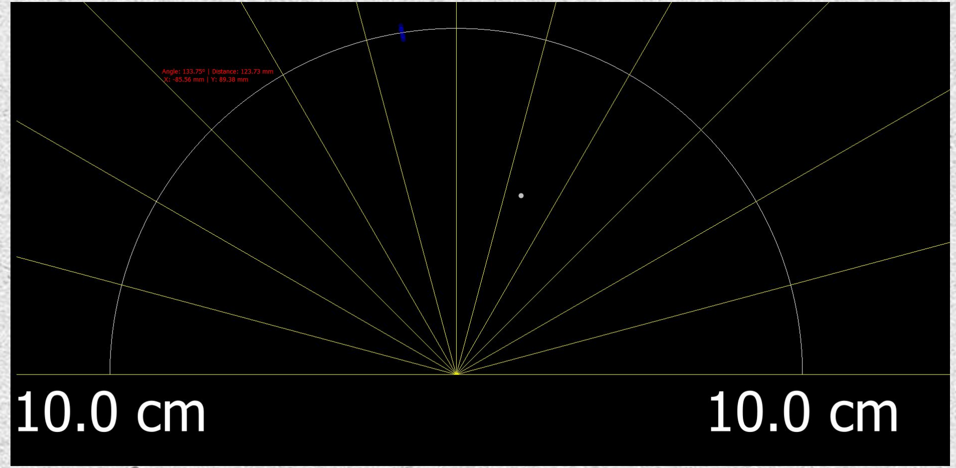


# Functional Requirements

- Transmit 40 kHz ultrasonic pulses in a phased array for detection
- Detects an object  up to 1 meter away and determine object direction using a phase delay with a phased array
- Amplify and filter incoming signals to reduce noise
- MCU converts time delay and phase shift data into an object's location and send that data to the Raspberry Pi over an MQTT connection
- Host a local web server using a Raspberry Pi to serve object detection data over Wifi to be visualized on a sweeping radar display

# Nonfunctional Requirements

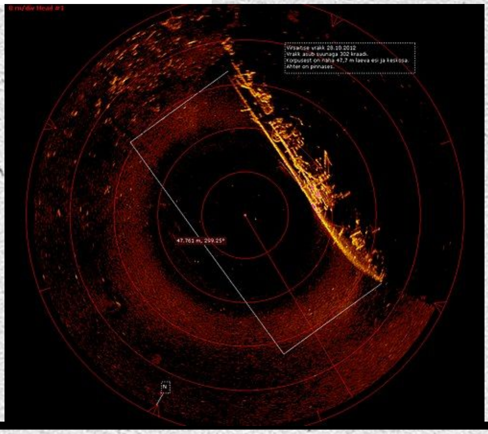
- **Have a clear and readable display (aesthetics)**
  - Cursor tracking
  - Share colors for object clusters
  - Zooming and Panning
- **Raspberry PI functioning as server for data points**
- **Linear phased array layout**



# Market/Literature Survey (Relevant Applications)

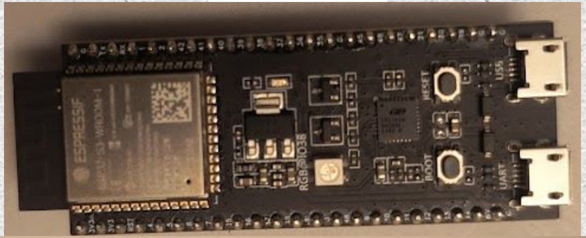
- Medical applications (scanning machinery)
- Nondestructive evaluation
- Sonar systems
- Security devices
- Proximity detection
- Water level monitoring

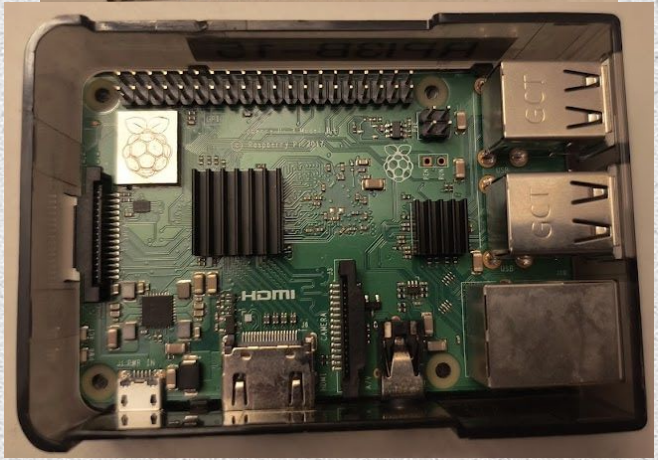




# Resource Requirements

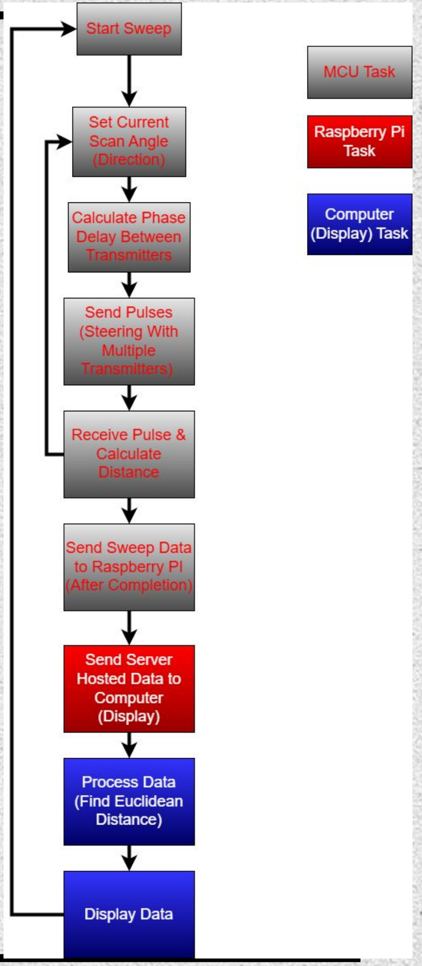
- MA40S4S/R (6S, 1R)
- ESP32 S3 Dev Kit (MCU)
- Adjustable Power Supply
- Raspberry PI 3b (As server for data pointers)
- 555 timers (for pulse generation)
- R/S Latches
- Filters





# Task Decomposition

- Configure phase delays for the transmitter array to steer the ultrasonic beam in specific directions using calculated time offsets
- Trigger 40 kHz ultrasonic pulses from the transmitters and receive echo signals using the receiver circuit
- Amplify and filter received signals to reduce noise
- Calculate time-of-flight between transmission and reception to determine the object distances
- Transmit object data location data to a Raspberry Pi using an MQTT connection
- Display transmitted data on a python-based GUI

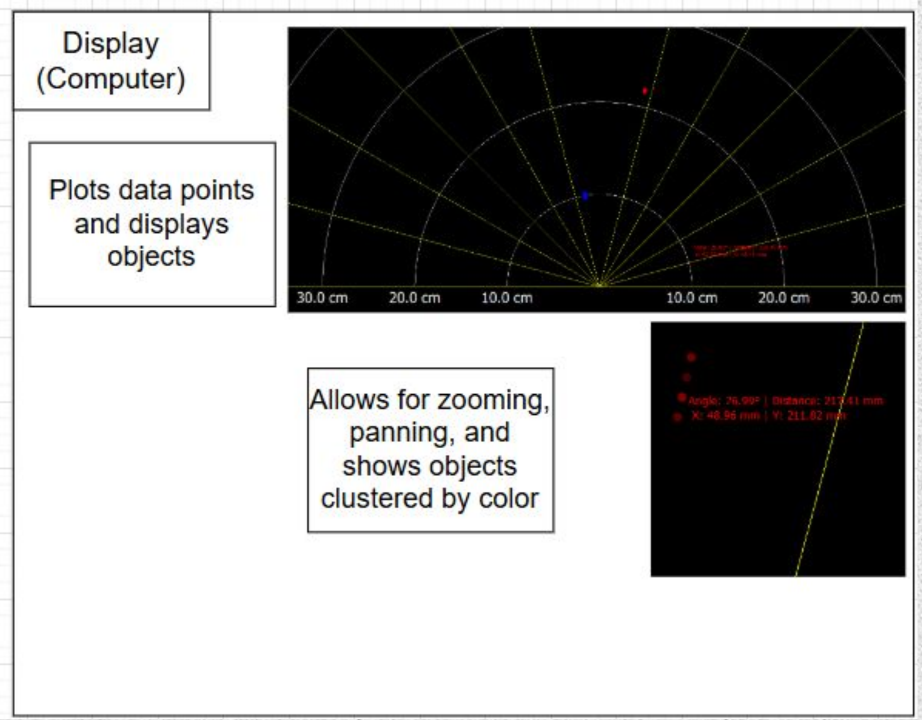


# Risk Identification and Mitigation

- **Transducer Protection**
  - The transducers are very sensitive and prone to damage
  - Ensure the transducers are powered according to specifications to avoid damage.
- **High Sound Intensity (Risk of hearing damage)**
  - Transducers will receive a reduced voltage (12v instead of 20v).
  - Hearing protection is required when within 3 meters of the device.
- **Voltage Step-up and Step-down (From and To MCU)**
  - Voltage to receiver should receive voltage above 3.3 volts (step down)
  - Voltage must be stepped-up to power the 555 timers
- **WIFI Interference**
  - Using ADC_1 channel rather than ADC_2

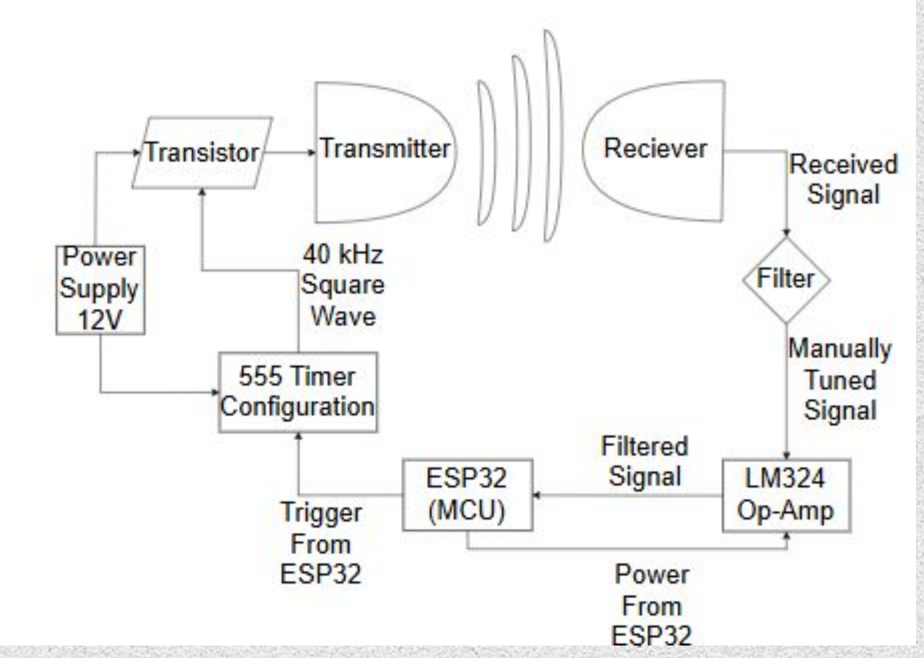
# Detailed Design (Display)

- **Display Receives Sweep Data from Server.**
- **Each reading is represented with a point on the display.**
    - Points fade overtime
- **Points from the same object are displayed with the same color.**
    - Clustered based on 1 cm proximity
- **The display allows for Zooming and Panning, and Cursor Tracking**



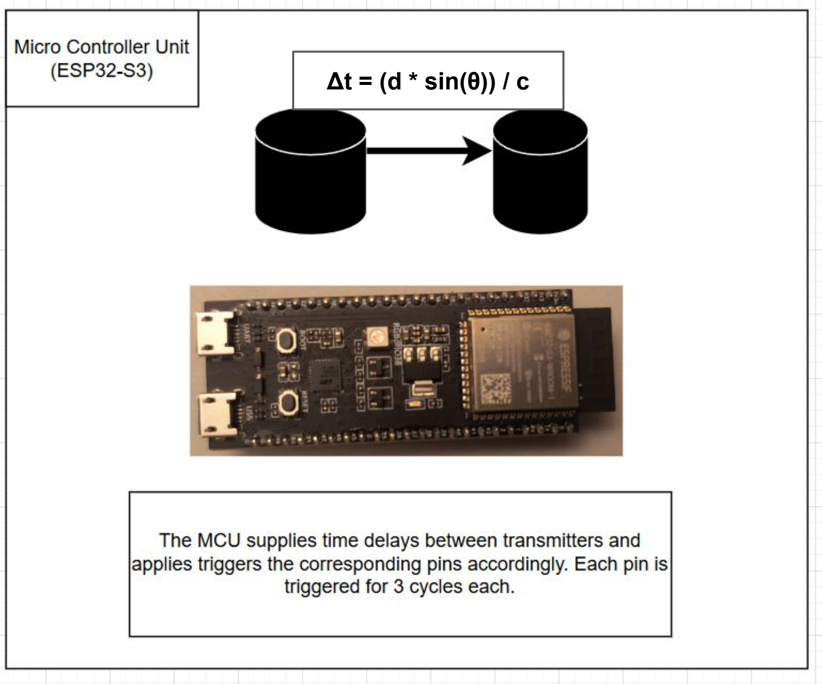
# Detailed Design (Hardware)

- Signals from the MCU sent from the transmitter pins (after a step-up) to fire the 555 timers.
- 555 timers send 40 kHz waves to which are further stepped.
- 40 kHz waves power the transmitters to produce a pulse.
- The signal of from a returning wave (object detected) is sent back to the receiver.
- The signal is filtered and sent back to the MCU for processing.



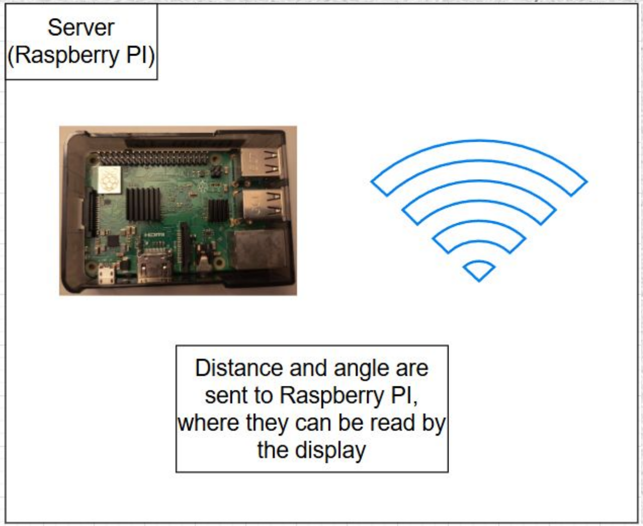
# Detailed Design (MCU)

- **Time delay formula: Δt = (d * sin(θ)) / c**
  - Each transmitter pin is triggered based on the time delay formula
  - If firing left (start delays from the right), and vice-versa for right.
- **MCU receives voltage values from waves received by the receiver**
- **Sends data to the Raspberry Pi to post on the web server**

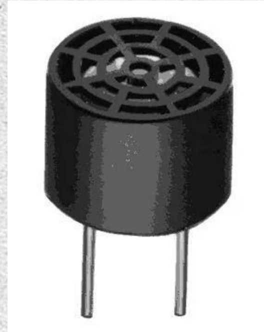


# Detailed Design (Server)

- **Server hosting**
  - Hosted locally on Raspberry Pi
  - Stores data from scan for display to use
  - Created using Apache
- **Server communication**
  - MQTT protocol allows communication between devices on same network
  - Setup using Mosquitto



# Design Tradeoffs



- **Transducers - MA40S4S/R**
  - Pros:  cost, 40 kHz signals
  - Cons: size, noise, hard to control phase
- **Microcontroller - ESP32-S3-DevKitC-1-N8R8**
  - Pros: processing power, WiFi connectivity
  - Cons: trouble with ADC, more expensive than previous MCUs

# Problems

# Core Challenges

- **Precise phase delay calculations**
  - Implement a phase delay control across a multi transmitter phased array to determine the scan direction
- **Accurate distance measurement**
  - Measuring time-of-flight using 40 kHz pulses, while filtering out external noises
- **Wireless data transmission**
  - Sending detection data from the ESP32 to the Raspberry Pi over Wifi using MQTT protocol

$\Delta t = (d*\sin(\theta)) / c$

$ToF = (2 * Distance) / c$

$Distance = (ToF/2) * c$

# Core Challenges Cont.

- **Web Server Hosting**
  - Hosted locally on the Raspberry Pi using Apache
- **Display**
  - Python used for improved functionality
    - Zooming, Panning, Cursor Tracking, Clustering
- **Large Transmitters (10 mm)**
  - Ideally the transmitters would be smaller to allow for critical spacing to be possible.
  - 10 mm is larger than the critical spacing, and greater than $\lambda/2$ which is the standard spacing.
  - Minimum beam width is ~5 degrees, which would need to be resolved with receiver triangulation.

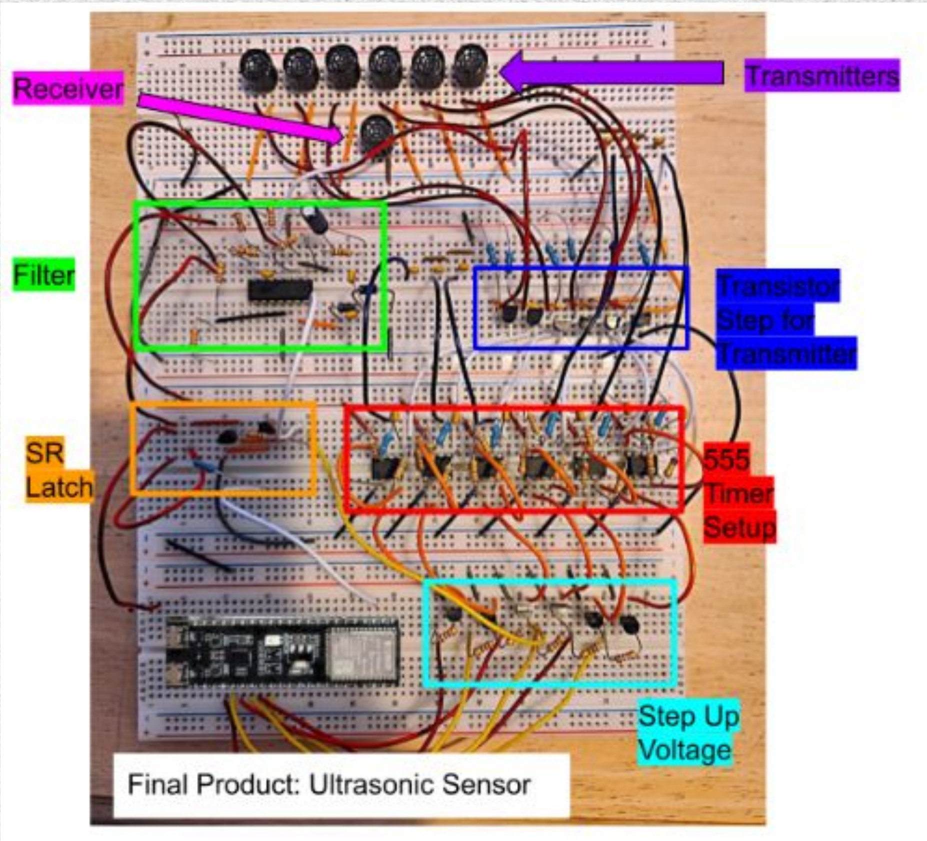
# Prototype

# Software

- **Python**
  - display and communication
- **Arduino IDE (C++)**
  - MCU programming
- **Apache**
  - Web server
- **Mosquitto**
  - MQTT
- **Falstad**
  - Circuit simulation

# Hardware

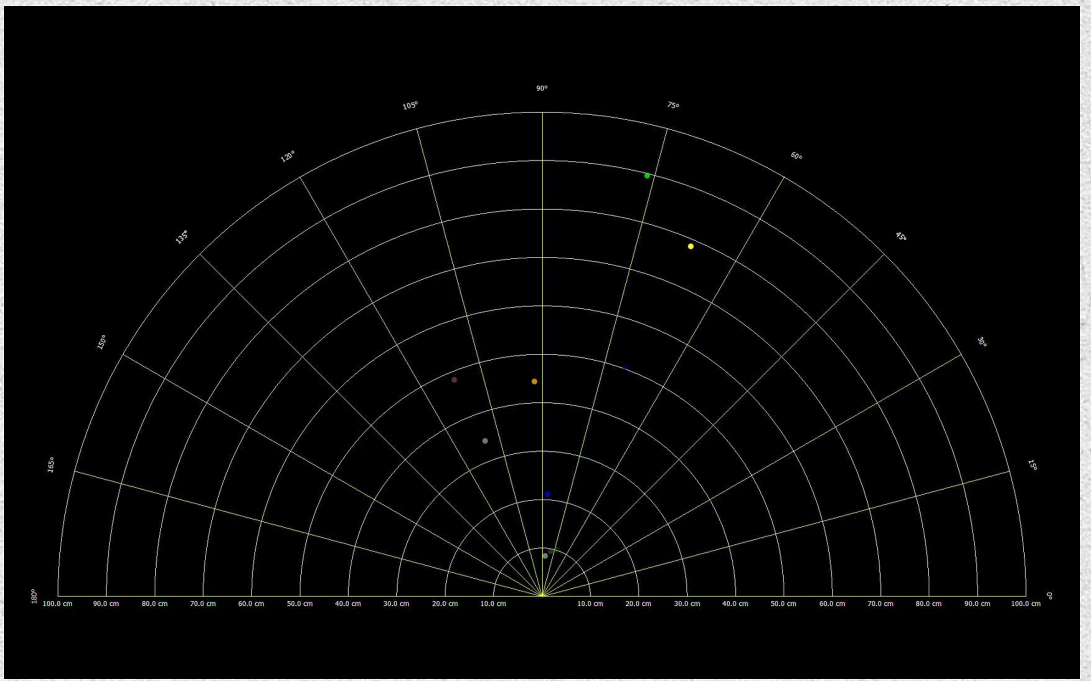
- 555 Timers
- Filter
- LM324 Op-Amp
- SR Latch



# Testing

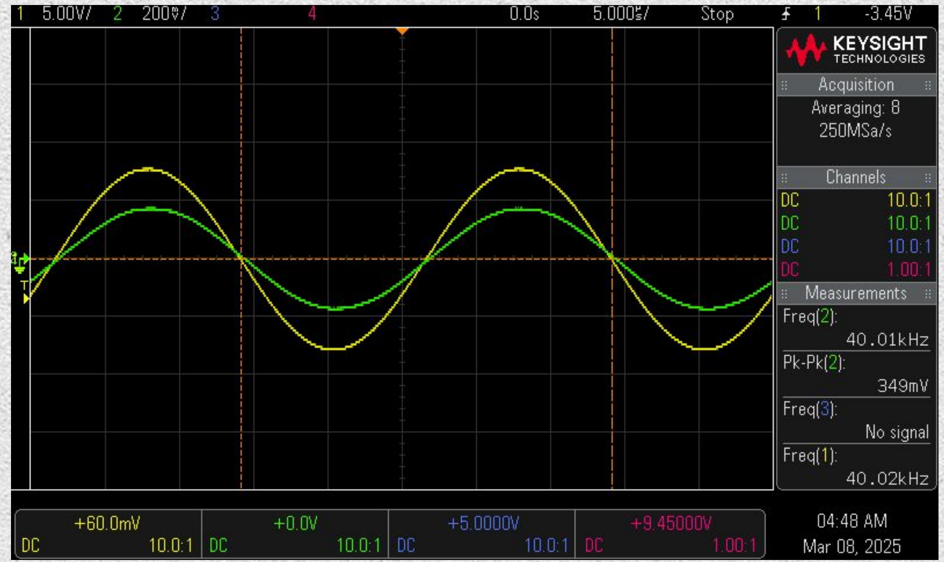
# Communication Between Components

- **Randomized points**
  - Sent from the ESP32 to the Raspberry Pi which was then plotted by the display



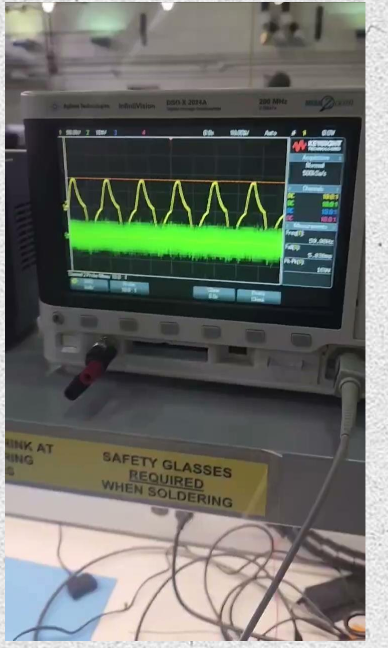
# Initial Hardware Testing

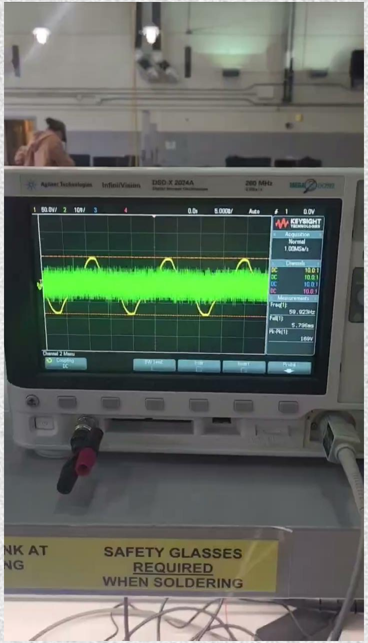
- **Sending a pulse directly to the receiver**
  - Pulse of transmitter and received signal at receiver align.
    - 555 timers were set up to generate pulse signals independently
    - Each timer was configured to send a pulse at specific intervals
    - Oscilloscope monitored the output waveform of the pulse signal



# Receiver Signal Testing

- When the transmitter is pulsing, the oscilloscope shows that the receiver can hear the transmitter
    - Oscilloscope displayed both the direct signal and reflected echo
    - Helped confirm the receiver was detecting transmitted and reflected waves



# Final Hardware Simulation

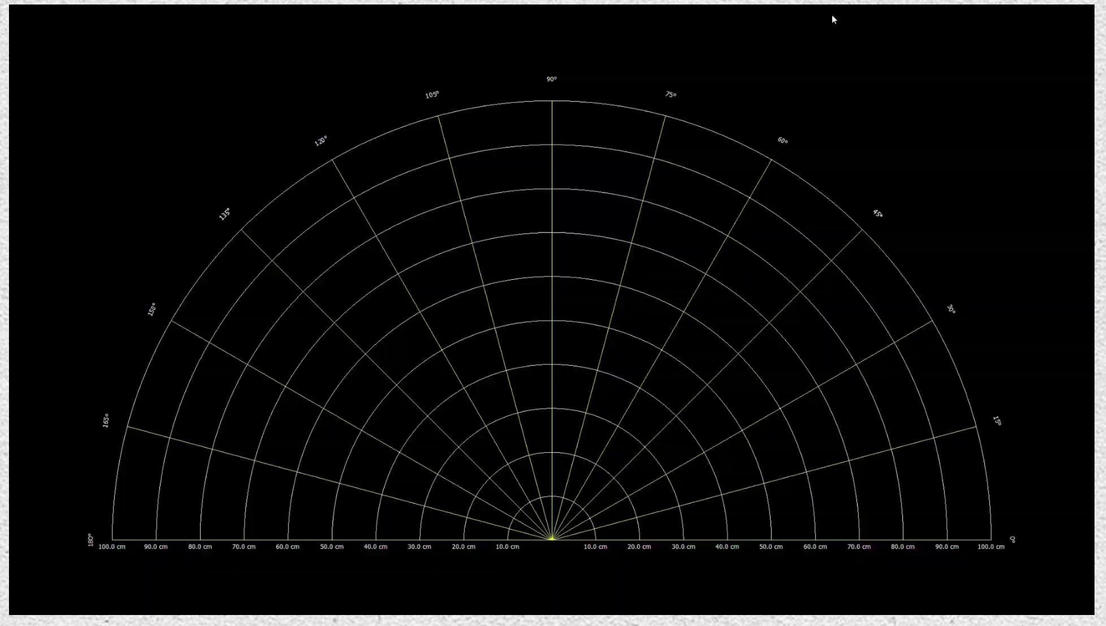
https://tinyurl.com/24x7d4ap

# Final Hardware Testing

- **Infeasibility**
  - Murata ultrasonic sensors **can not** steer beams electrically
  - Beam shaping through frequency reduces sound pressure
  - Object size/shape recognition needs arrays and complex processing
  - Not cost-effective or realistic for small-scale implementation
- **Testing vs. Simulation**
  - Physical tests showed significant deviation from simulation results
  - Echo noise, uncontrolled reflections, and dampening affected accuracy
  - Lack of beam control and inconsistent signal strength impacted detection
  - Environmental factors not accounted for in simulation

# Final Result

# Demo


90°
105°
75°
120°
60°
135°
45°
150°
30°
165°
15°
180°
0°
100.0 cm
90.0 cm
80.0 cm
70.0 cm
60.0 cm
50.0 cm
40.0 cm
30.0 cm
20.0 cm
10.0 cm
10.0 cm
20.0 cm
30.0 cm
40.0 cm
50.0 cm
60.0 cm
70.0 cm
80.0 cm
90.0 cm
100.0 cm

# Improvements Over Previous Years

- **Wireless communication**
- Hardware/Circuit
  - More complex design with improved signal filtering
  - Transducers now controlled by MCU with option for independent operation
  - Cleaner overall layout and wiring for better performance and debugging
- **Display Improvements**
  - Panning, Zooming, Cursor Tracking
  - Point Clustering
- **Smaller Transmitters (16 mm —> 10 mm)**
- **Circuit Simulation**
  - Precise 40 kHz pulses using 555 timers
  - Receiver values baseline set using SR latch
  - Ideal Environment that displays accurate readings

# Project Direction & Next Steps

- **Infeasibility**
  - Murata sensors can't steer beams; tuning reduces power
  - Object sizing needs sensor arrays and complex processing
    - **Dampening/Echo overlap** causes false positives.
  - Lack of proper testing tools for beam forms and pressure fields.
    - Suggested: **acoustic chamber, mic array, high-speed scope**
  - Changing transducer element type (currently  basic piezoelectric elements)
    - Arduino ultrasonic sensor
- **Faster  and more reliable computations**
  - Potential move to FPGA or quicker MCU
    - More precise time delays (nano seconds)
  - MCU with effective internal pulldown for ADC channels

# Project Direction & Next Steps Cont.

- **Phased array design changes**
  - Additional transmitters
  - Square or 2D array
  - Additional receivers for triangulation
  - Smaller elements (if possible)

# Questions?

# Image Sources

- https://nikeson.com/en-us/products/ultrasonic-level-sensor
- https://us.medical.canon/products/ultrasound/
- https://en.wikipedia.org/wiki/Sonar
- https://en.m.wikipedia.org/wiki/File:Python-logo-notext.svg
- https://en.m.wikipedia.org/wiki/File:Arduino_Logo.svg
- https://commons.wikimedia.org/wiki/File:Apache_HTTP_server_logo_%282019-present%29.svg