4.2 Design Exploration

4.2.1 Design Decisions

One key decision that was made was to use a Raspberry Pi for its cost-effectiveness and functionality. It was also chosen because of its ability to communicate with the microcontroller and processing IDE. Because it is able to process real-time data, our radar display will be more precise. We also wanted a wireless connection for this project, and the Raspberry Pi will allow us to do that.

Deciding which transducer to use was another key decision. Choosing the transducer was one of the most important tasks for this project because they are the parts that are required for sending and receiving ultrasonic pulses. We had to consider the size and range of the transducer as we were looking for transducers that were around 10 mm in diameter and a range of 1 meter. The MA40S4S/R transducers were chosen because they had the criteria requiring 10 mm diameter transducers with a range of at least 1 meter.

Another important key decision that was made for this project was determining which Microcontroller Unit to use for the radar's data processing. The past implementations used ESP32 MCUs, so this was the general family of products we searched for. The most recent implementation used the ESP32 D1, which is fairly outdated, and they expressed that more processing power could be a good place for improvement. We began looking into the C6 and S3 models and found that the C6 models emphasize power use reduction rather than processing power like the S3. We ended up further exploring the S3 models and found a model with 8 MB of flash and 8 MB of PSRAM. The PSRAM should provide a buffer for calculations, which should prove helpful in our project and should allow for a reduction in delays in displaying data.

4.2.2 Ideation

When choosing our transducers, we had many possibilities. We chose the MA40S4/R, with some of our other considerations included below.

- TCT40-16R/T
    - This transducer is 16mm in diameter; we chose not to pick it because of its size.
- CUSA-T60-150-2400-TH
    - This transducer is 14mm in diameter and costs $2.95; despite its cheap cost, we wanted a smaller transducer.
- MCUSD14A40S09RS
    - This transducer is 14mm in diameter; while it is smaller than the previous option, we still wanted a transducer that was smaller
- 400SR16
    - This transducer has a diameter of 13mm, and we chose not to pick this because of its high cost, being around $8.5
- Eu10POF40H07T/R
    - This transducer has a diameter of 10mm, which is the same as the one we chose, but costs $9.95 each, which is about double the cost of the transducer we chose.

We ultimately picked the MA40S4S/R because of its relatively small diameter of 10mm and cheap cost of $49.30 for 10 transmitters, each costing $4.93 and the receiver costing $5.49.

4.2.3 Decision-Making and Trade-Off

Demonstrate the process you used to identify the pros and cons of trade-offs between each of your ideated options. You may wish to include a weighted decision matrix or other relevant tool. Describe the option you chose and why you chose it.

Transducer Criteria – Chosen: MA40S4S/R

- The transducers should be as small as possible (with 10 mm diameter being the expected size).
- The transducers should have a range of at least one meter.
- The transducers should be the cheapest option, fulfilling all other criteria.
- The transducers should have a 40KHZ signal.
- The transducers should have a wide viewing range.
- The transducers should not have excessive decibel production at far ranges (outside of the detection area of one meter)

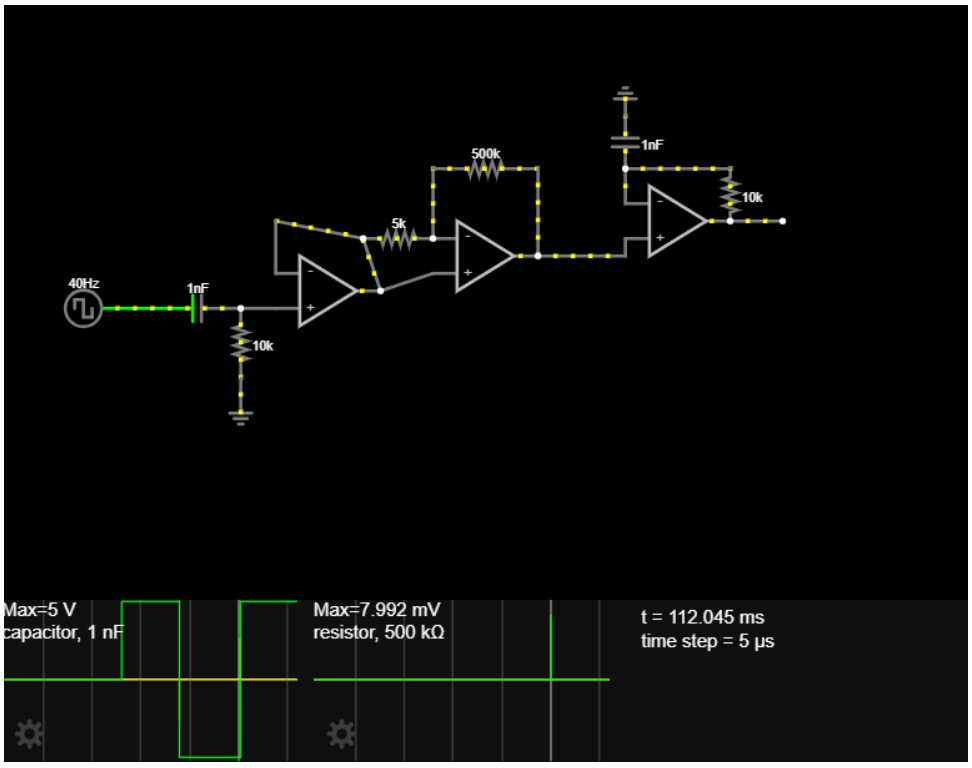
Microcontroller Unit Criteria – Chosen: ESP32-S3-DevKitC-1-N8R8

- Provides ample processing power (significantly more powerful than the D1 of past implementations).
- It must have a small form factor to easily fit on the PCB board for the final design.
- It must have a clock rate of at least 240MHZ.
- It must have at least 8MB of flash memory.

Wireless Data Transmission – Chosen Raspberry Pi 3b

- The device must be within a reasonable price range (Less than $50).
- The device must allow for a 2.4 GHZ wireless band.
- The device must allow for easy connection of the computer for display processing.

4.3 Proposed Design



(Figure above is the receiving portion)

4.3.1 Overview

The ultrasonic radar project is a detection system that uses sound waves to pinpoint an object's location and distance from the sensor. It works by sending out ultrasonic waves and then listening for echoes that bounce off objects. We can figure out an object's location by recording the time it takes for the echo to return.

Key Components

- Raspberry Pi - The Raspberry Pi receives data from the microcontroller and our display is able to receive the same information.
- Transducers (Transmitter and Receiver) are how the radar sends and receives the sound waves. The transducers will emit sound waves, and if the waves reflect and are received, then an object is detected, and the data is relayed to the Raspberry Pi.
- Microcontroller - The microcontroller is a middleman between the transducers and Raspberry Pi. It manages the timing of the signals sent by the transducers. It is also responsible for sending the echo data to the Raspberry Pi.
- Display - Show the data received from the Raspberry Pi and accurately depict objects that were detected. Our display will also have to process the data received from the Raspberry Pi.

4.3.2 Detailed Design and Visual(s)

The ultrasonic radar project is a Raspberry Pi-based detection system designed to measure distances and detect objects through ultrasonic waves. The core components of this system include the Raspberry Pi microcontroller, MA40S4S/R ultrasonic transducers, signal amplification and processing circuitry, a display interface, and power management circuitry. The Raspberry Pi is the central processor responsible for managing pulse timing, handling data processing, and updating the display. A Python script on the Raspberry Pi generates a pulse to activate the transducer, then switches to a listening mode to detect the echo return. By measuring the time delay between sending the pulse and receiving the echo, it calculates object distance using the speed of sound.

The MA40S4/R transducer, optimized for 40 kHz operation, serves as both the emitter and receiver of ultrasonic waves. When triggered by the Raspberry Pi, it sends out a pulse; if the waves encounter an object, they reflect back to the transducer, which then sends the received signal to the Raspberry Pi. An amplification and filtering circuit amplifies the signal and reduces ambient noise to ensure these return signals are processed accurately. The amplifier circuit, typically an op-amp configured in a non-inverting layout, enhances weak return signals, while low-pass filters remove unwanted high-frequency noise that can interfere with detection accuracy. The Raspberry Pi then uses this data to update the display interface, an LED array or an LCD, to indicate detected objects and visually display approximate distances in real time. Finally, the power management circuitry supplies stable voltage across all components, ensuring consistent performance. By integrating these systems, the Raspberry Pi radar effectively detects and displays objects within a specified range.

4.3.3 Functionality

In its real-world context, the ultrasonic sensor system is designed to detect objects within a specific range and provide feedback based on the proximity and size of those objects. The user begins by placing the device in an environment where monitoring is required—such as a lab setup or mounted in a vehicle for obstacle detection. After positioning, the user powers on the device, allowing it to initialize the radar sensor and signal processing circuits. Once active, the user initiates the scanning process by pressing a button on the device or remotely controlling it through a paired interface, such as a mobile app or computer. The sensor system then emits ultrasonic pulses through the MA40S4/R transducer. As these pulses encounter

objects, they reflect back, and the device measures the time taken for the echoes to return. This timing data enables the system to calculate the distance and, potentially, the size of any detected objects.

In response to the calculated distance, the system provides feedback to the user. For example, if an object is detected within a critical range, the device can alert the user through audio or visual cues or by sending a notification to the interface. The system continues to scan until it is manually deactivated or a specified object is identified, making it suitable for real-time tracking or monitoring applications. The system automatically adjusts pulse-echo timing based on environmental conditions, such as background noise or clutter, enhancing signal amplification when necessary to optimize range and accuracy. Detected data is stored or transmitted to an external device if required, allowing for distance logging over time for further analysis. A visual representation could illustrate each step: the user activates the system, the device sends a pulse, an object reflects the echo, the system processes the data, and feedback is provided based on the distance detected.

4.3.4 Areas of Concern and Development

The current design of the ultrasonic sensor system is well on its way to meeting the requirements, but several concerns need to be addressed to ensure accuracy and proper functionality in real-world conditions. The system aims to detect objects based on their proximity using ultrasonic pulses, providing feedback to the user based on the detected range. The Raspberry Pi processes data from the MA40S4S/R transducer, which is then amplified and filtered by a custom circuit. However, key issues must be considered in refining the system to fully meet user and client expectations.

1. **Primary Concerns:**

Accuracy: Object detection accuracy depends on the transducer's ability to send and receive pulses effectively. External factors like noise and object shape can cause measurement errors, especially in high-interference environments. Improved calibration and additional filtering or dual sensors may help address this.

Signal Processing: The Raspberry Pi may not process signals fast enough for real-time applications, leading to delays or inaccuracies, particularly in dynamic environments. A microcontroller or DSP could improve real-time performance but would increase cost and complexity.

User Interface and Display: The system's effectiveness depends on how clearly the data is presented. Testing different display methods, such as visual or auditory feedback, is necessary to ensure clarity and usability.

Environmental Variability: Environmental factors like temperature and humidity can affect sound wave propagation, leading to accuracy issues. Regular calibration and possible environmental sensors can help improve performance in various conditions.

2. **Immediate Plans to Address Concerns:**

Enhanced Calibration: WE plan to refine the calibration process to better account for environmental factors and ensure accurate readings. I will test the system in different environments to understand how these factors impact performance.

Improving Signal Processing: We are considering integrating a microcontroller or DSP to offload some of the real-time signal processing from the Raspberry Pi. This would improve the responsiveness and accuracy of the system. Alternatively, optimizing the existing Raspberry Pi code to handle processing more efficiently could be a first step.

User Interface Testing: We will design and test different display formats to determine the most intuitive and effective way to present data to the user. This might include graphical displays or more basic feedback, such as LED indicators for proximity warnings.

Environmental Testing: We plan to conduct tests in various environmental conditions (different temperatures, humidity levels, and materials) to determine how sensitive the system is to these variables. This will help identify where adjustments or compensations are needed.

3. **Questions for Clients, TAs, and Faculty Advisors:**

For the Clients: Are there specific environments where this sensor system will be used, and how critical is the system's performance in those environments (e.g., varying temperature, humidity, or interference levels)?

For the TAs: Do you have suggestions for optimizing the Raspberry Pi's signal processing capabilities for real-time applications, or would integrating a microcontroller be a better route?

For Faculty Advisors: Are there alternative sensor technologies or enhancements (such as dual-transducer systems or frequency modulation) that could improve accuracy without significantly increasing cost or complexity? How can we best test and validate the system's accuracy under varying conditions?

4.4 Technology Considerations

MA40S4S/R Transducer:

- Strengths:
  - Reliable and accurate for short/medium-range detection; effective in controlled environments.
- Weaknesses:
  - Limited performance in high-noise environments or with irregularly shaped objects.
- Trade-offs:
  - Cost-effective but limited in adaptability to complex or noisy environments.
- Alternatives:
  - Dual-transducer systems or adaptive frequency modulation sensors, although they would increase cost and power consumption.

Raspberry Pi:

- Strengths:
  - Flexible, supports various programming languages, handles data processing and user interface integration.
- Weaknesses:
  - Limited processing speed and not optimized for real-time signal processing.
- Trade-offs:
  - Offers flexibility in programming but may introduce delays in high-frequency applications.
- Alternatives:
  - Dedicated microcontrollers or digital signal processors (DSPs) for faster real-time processing, but with reduced programming flexibility.

Custom Amplification and Filtering Circuit:

- Strengths:
  - Customizable to specific detection needs; boosts signal range and filters noise.
- Weaknesses:
  - Time-consuming to fine-tune; sensitive to environmental factors like temperature and humidity.
- Trade-offs:
  - Balances sensitivity and noise reduction but may risk false positives.
- Alternatives:

    - Pre-built amplifiers with adaptive filtering capabilities could complicate integration and add system complexity.

4.5 Design Analysis

Based on the testing we have done, all of our parts have worked on their own.

- MA40S4S/R
    - We have tested both the senders and the receiver, which have functioned as expected.
- Raspberry Pi 3B
    - It was able to host the web server locally.
- Display
    - We are using the processing IDE to display the location of the objects; we have tested our display using mock data and have gotten the results to display.

In the future, we will need to start combining our parts and getting everything to work smoothly together. As far as we are aware, the proposed design will work. We feel that this design is feasible in the limited time we have to complete this project.