EEE3097S 2023 ASSIGNMENT 1: PAPER DESIGN
Acoustic Triangulation

Joshua King: KNGJOS007

Dante Webber: WBBDAN003

Michael Awe: AWXMIC001

Contents

Individual contributions

Joshua King: KNGJOS007 – Introduction, requirements and sub-systems Pi Synchronization and Signal Acquisition

Dante Webber: WBBDAN003 – Testing, Development Timeline and subsystems time delay estimation and GUI

Michael Awe: AWXMIC001 – Sub-system requirement and specifications and subsystems Triangulation and Power

1 Introduction

This report marks the commencement of our project focused on Acoustic Triangulation using Time Difference of Arrival (TDoA) within a distributed sensor network. Our objective is to design and implement a system capable of accurately locating a sound source within a defined grid in x y-coordinates. By leveraging the principles of TDoA and the capabilities of Raspberry Pi microcontrollers, we aim to create a robust solution that not only accurately triangulates the sound source's position but also accommodates sound sources of different frequencies. This versatility opens the door to diverse applications and potential advancements in fields where precise acoustic source localization is essential.

2 System Level Requirements

- **Positional Accuracy using X-Y Coordinates:** The system shall employ X-Y coordinates for accurate localization of sound sources within the designated grid area.
- **Microphone Array:** The system shall incorporate a four-microphone array to facilitate comprehensive sound sensing.
- **Comprehensive Audio Signal Simulation:** The system shall generate simulated audio signals, varying in positions and subjected to diverse environmental conditions, to facilitate thorough testing.
- **Modular Subsystem Architecture:** The system shall be organized into distinct subsystems, ensuring efficient operation and effective system management.
- **A1 Paper Grid Confinement:** The system’s operational scope shall be limited exclusively to a predefined A1 paper grid.
- **Raspberry Pi Zero Utilization:** The system shall leverage the capabilities of two Raspberry Pi Zero microcontrollers for seamless implementation.
- **Project Completion by End of Term:** The system development, implementation, and testing shall be completed by the end of the term.
- **Sound Source Localization with Different Frequencies:** The system should be able to accurately determine the (x, y) coordinates of sound sources emitting different frequencies within the established grid boundaries.
- **Verification of System Robustness:** The system shall undergo rigorous testing to verify its robustness, and accuracy of sound source localization across varying frequencies.

3 Subsystem analysis & design considerations

3.1 Pi Synchronization

Comparison of Possible Implementations:

- Possible methods for Pi synchronization include using Shared Network Time Protocols (NTP), Hardware-Based Synchronization or Software-Based Synchronization via clock signals.
-

Feasibility Analysis:

- NTP synchronization can provide precise time alignment if network stability is maintained.
- Hardware-based synchronization would be feasible with GPIO pins or external clock sources.
- Software-based synchronization would be feasible using signal pulses exchanged between Raspberry Pis.

Possible Bottlenecks:

- Network connectivity issues could impact the accuracy of NTP synchronization.
- Hardware-based synchronization requires additional components and would be prone to hardware limitations.
- Software-based synchronization could face challenges in maintaining consistent timing accuracy.

3.2 Signal Acquisition

Comparison of Possible Implementations:

- Possible methods for signal acquisition include:
    - *Synchronous Sampling across Microphones:* Simultaneously capturing signals from multiple microphones, allowing analysis and processing of coherent data.
    - *Asynchronous Sampling with Cross-Correlation:* Collecting signals from microphones at different times and using cross-correlation to find the time delay between them, enabling alignment for analysis.

Feasibility Analysis:

- Synchronous sampling would be feasible with precise clock synchronization.
- Asynchronous sampling would be feasible if cross-correlation techniques were effectively implemented.

Possible Bottlenecks:

- Synchronous sampling would face challenges in maintaining strict timing alignment.
- Asynchronous sampling would require complex signal processing algorithms to achieve accurate cross-correlation.

3.3 Time Delay Estimation

Comparison of Possible Implementations:

- **Methods for time delay estimation include:**
    - ***Cross-Correlation:*** Cross-correlation involves comparing two signals by sliding one over the other and calculating the similarity at different positions. The position that maximizes this similarity corresponds to the time delay between the signals.
    - ***Generalized Cross-Correlation (GCC):*** GCC is an enhanced version of cross-correlation, specifically designed for signals in different environments. It considers factors like phase differences, improving accuracy in estimating time delays.
    - ***High-Resolution Techniques (MUSIC):*** Multiple Signal Classification (MUSIC) is a high-resolution method for estimating time delays. It utilizes spectral analysis to distinguish signals with slight delays, enabling precise measurement even when signals are closely spaced in time.

Feasibility Analysis:

- Cross-correlation is feasible with real-time signal processing.
- GCC and MUSIC techniques are feasible if computational resources are sufficient.

Possible Bottlenecks:

- Cross-correlation would struggle with noise and might require additional noise reduction methods.
- GCC and MUSIC techniques would require significant computational resources, impacting real-time performance.

3.4 Triangulation

Comparison of Possible Implementations:

- **Triangulation methods include**:
    - ***Centroid Localization:*** Determining the position by calculating the center point of intersecting lines from multiple reference points.

- ***Multilateration:*** Estimating location by measuring distances from multiple known reference points and intersecting the resulting spheres.
- ***Least Squares Estimation:*** Finding the most likely position by minimizing the sum of squared differences between calculated and actual distances from reference points.

Feasibility Analysis:

- Centroid localization is feasible as a simple approach.
- Multilateration and least squares estimation are feasible if implemented with accurate TDoA values.

Possible Bottlenecks:

- Centroid localization might lack the accuracy required for precise sound source localization.
- Multilateration and least squares estimation would be sensitive to errors in TDoA calculations.

3.5 User Interface (GUI):

Comparison of Possible Implementations:

- Development of a standalone graphical application using libraries like PyQt or Tkinter.
- Integration of a web-based interface using HTML, CSS and JavaScript frameworks.

Feasibility Analysis:

- Standalone application feasible with appropriate GUI libraries and programming skills.
- Web-based interface feasible with knowledge of web development tools and frameworks.

Possible Bottlenecks:

- Developing a standalone application would require a learning curve for GUI library usage.
- Web-based interfaces would be subject to browser compatibility challenges.

4 Subsystem & Sub-subsystems requirements & specifications

4.1 Pi Synchronization

Subsystem Requirements:

- The Raspberry Pi microcontrollers should be synchronized to ensure accurate time measurements for sound signals.
- The synchronization method should minimize the time delay between the two Raspberry Pis.
- Synchronization should be initiated at system startup and maintained during operation.

Subsystem Specifications:

- The Raspberry Pi microcontrollers should be synchronized using the Network Time Protocol (NTP) or Precision Time Protocol (PTP) to achieve accurate system time alignment.
- The synchronization error between the Raspberry Pis should be within ±1 millisecond.
- The system should be able to maintain synchronization during system operation, even in the presence of network interruptions.
- The synchronization process should be automated and initiated upon system startup.

4.2 Signal Acquisition

Subsystem Requirements:

- The system shall incorporate a four-microphone array utilizing Adafruit I2S MEMS microphone breakout boards to facilitate comprehensive sound sensing.
- The system should be capable of capturing high-quality audio signals from the four MEMS microphones simultaneously.
- The microphones should be connected to the Raspberry Pis via a suitable protocol.
- The sampling frequency of the audio signals should be sufficient for accurate time delay calculations.
- The captured signals should be stored in suitable data structures for further processing.

Subsystem Specifications

- The Adafruit I2S MEMS microphone breakout boards should be provided with a 3.3V power supply.
- The microphones should be connected to the Raspberry Pis using the I2S communication protocol.
- The sampling frequency for audio signals should be at least 44.1 kHz to capture adequate frequency information.

- The capture resolution should be 16 bits per sample to ensure sufficient dynamic range.
- The captured audio signals should be stored in buffers for real-time processing.

4.3 Time Delay Estimation

Subsystem Requirements

- The system should accurately measure the time differences of arrival (TDoA) of the sound signals at each microphone.
- Time delay estimation should account for any synchronization error between the Raspberry Pis.
- The time delay estimation method should be robust against environmental noise and reflections.
- The estimated time delays should be converted into distance differences using the speed of sound.

Subsystem Specifications

- Time delay estimation should be based on cross-correlation analysis of the audio signals received from different microphones.
- The time delay estimation should account for any phase shifts induced by hardware or signal processing.

4.4 Triangulation

Subsystem Requirements

- The system should use the estimated distance differences to locate the position of the sound source within the rectangular grid.
- Triangulation should account for the known coordinates of the microphones and their corresponding distance differences.
- The triangulation algorithm should handle different grid sizes and adjust accordingly.
- The computed two-dimensional position coordinates should be accurate and precise.

Subsystem Specifications

- The triangulation algorithm should use trilateration.
- The triangulation algorithm should account for the propagation speed of sound, which is approximately 343 meters per second.
- The algorithm should handle grid sizes up to A1 dimensions (841 mm × 1189 mm) to ensure scalability.

- Interpolation or optimization techniques, such as least squares, should be employed to improve the precision of position estimation.

4.5 User Interface

Subsystem Requirements

- The system should provide a graphical user interface (GUI) to display the predicted results.
- The GUI should show the real-time position of the sound source on the printed grid.
- The GUI should be easy to understand and interact with.
- The system should provide options to calibrate and configure the system through the GUI.

Subsystem Specifications

- The graphical user interface should be developed using Python and a GUI framework such as Tkinter.
- The interface should display the real-time position of the sound source on the printed grid.

4.6 Power

Subsystem Requirements

- The system should be designed to operate using the power supply provided to the Raspberry Pis.
- Power consumption should be minimized to ensure practicality and longevity of the system.

Subsystem Specifications

- The Power submodule must provide the Raspberry Pi Zero's with a 5V supply.

5 Inter-subsystem Interactions

5.1 Pi Synchronization

- a. The Pi Synchronization subsystem interacts with the Time Delay Estimation subsystem to ensure that accurate time measurements are used for TDoA calculations.
- b. The Pi Synchronization subsystem provides synchronized timestamps to the Signal Acquisition subsystem for accurate timing of audio signal capture.
- c. The Triangulation subsystem relies on synchronized timestamps from the Pi Synchronization subsystem to calculate precise sound source positions within the grid.

5.2 Signal Acquisition

- a. The Signal Acquisition subsystem interacts with the Pi Synchronization subsystem to receive synchronized timestamps for capturing audio signals.
- b. The Signal Acquisition subsystem provides captured audio signals to the Time Delay Estimation subsystem for cross-correlation analysis.
- c. The User Interface subsystem may request access to the captured audio signals for monitoring or analysis purposes.

5.3 Time Delay Estimation

- a. The Time Delay Estimation subsystem depends on the Signal Acquisition subsystem to provide audio signals captured by the microphones for cross-correlation analysis.
- b. The Time Delay Estimation subsystem calculates time delays between audio signals received at different microphones.
- c. The Triangulation subsystem receives the estimated time delays from the Time Delay Estimation subsystem for further processing.

5.4 Triangulation

- a. The Triangulation subsystem utilizes the estimated time delays from the Time Delay Estimation subsystem along with known microphone coordinates to determine the sound source position within the grid.
- b. The Triangulation subsystem updates the sound source position in real-time and provides the coordinates to the User Interface subsystem for display on the graphical user interface.

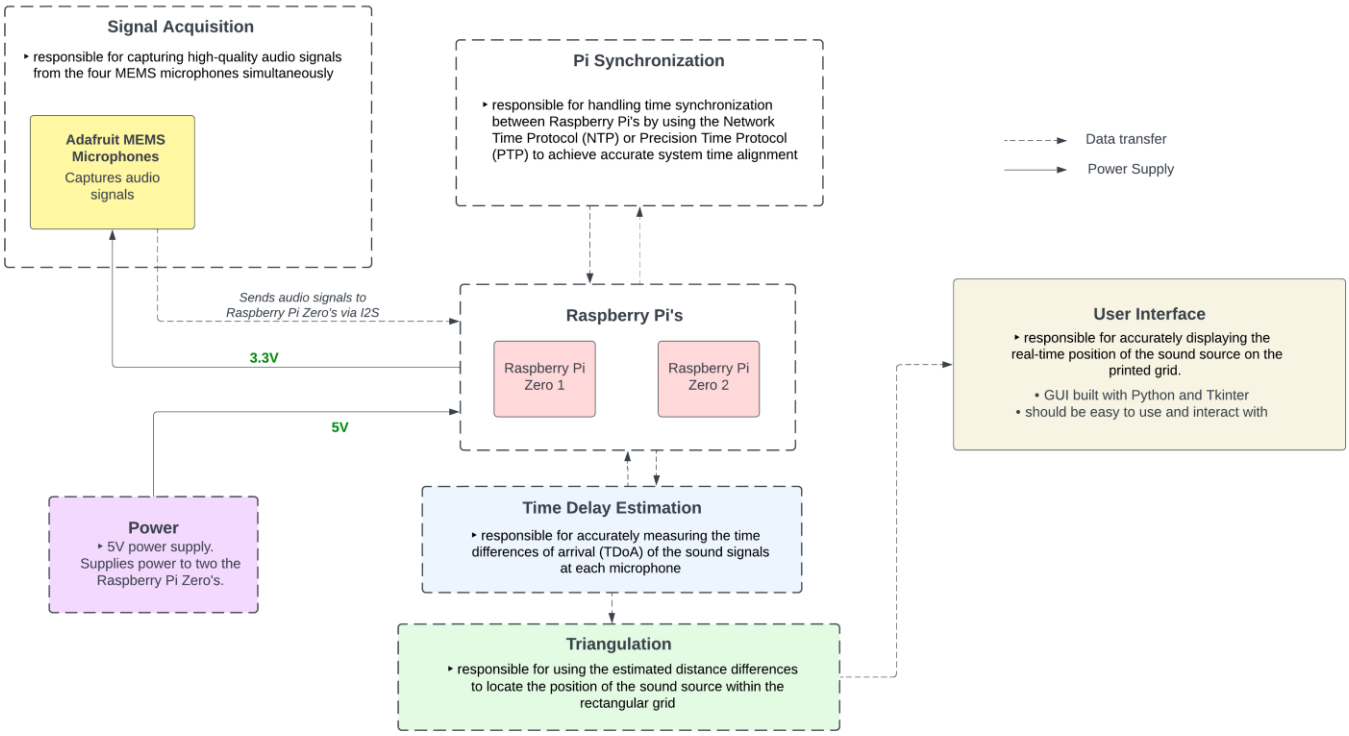
5.5 User Interface

- a. The User Interface subsystem interacts with the Triangulation subsystem to receive updated sound source position coordinates.
- b. The User Interface subsystem displays the sound source position in real-time on the graphical user interface.
- c. The User Interface subsystem provides options to calibrate the system and configure various parameters of the system.
- d. The User Interface subsystem may request access to captured audio signals from the Signal Acquisition subsystem for monitoring or analysis purposes.

5.6 Power

- a. The Power subsystem interacts with all other subsystems to ensure proper power management, startup, and shutdown sequences.
- b. The Power subsystem provides the necessary power supply to the Raspberry Pi microcontrollers and other components.
- c. The Power subsystem may receive signals or requests from the User Interface subsystem to initiate power-related actions, such as entering a low-power mode or initiating shutdown.

Subsystem Block Diagram



6 Acceptance Test Procedures

The success of each sub-system and the main overall system will be determined by whether the acceptance test procedures defined in this section are met. Detailed testing and success criteria for each sub-system, as well as expected results will be discussed in this section.

All the tests that involve audio signals should be run with as little other noise as possible to isolate the sound source. Only the final noise robustness test should be run with external noise added to the system.

6.1 Pi Synchronization

To test that the Raspberry Pis are successfully synchronized, a loop time recording function will be used. This function will contain a loop that stores the current time in an array, repeating 20 times with a delay of 2 ms between each execution.

- Run the function on both Raspberry Pis simultaneously.
- Compare the data in the array from each Raspberry Pi and if the times match, then the Pis are successfully synchronized.

6.2 Signal Acquisition

To test the quality of the microphones, they must first be tested individually, and then simultaneously. Before the individual and simultaneous tests, it is assumed that the Raspberry Pi's are powered and have been programmed to receive and store data from the Adafruit I2S MEMS microphone breakout boards.

6.2.1 Individual Microphone Board Test

Each Adafruit I2S MEMS microphone breakout board will go through a set of tests to test the functionality of the sub-system. The following tests shall be run in sequence:

- Connect the microphone breakout board to the Raspberry Pi and test that it is recognized.
- Place the A1 paper grid down in a controlled space and do not move it for the remainder of the tests.
- Once recognition is confirmed, place the sound source and the microphone board on adjacent grid points. Hereafter do not move the sound source. If, due to unpredictable circumstances, it is moved, ensure that it is placed back on the same grid point as before.
- Prepare a short audio signal (10s long or less) on the sound source. This audio signal must be a completely known, well-defined and simple audio signal.
- With both the source and the microphone board placed securely, start the microphone recording and use the sound source to generate the audio signal.
- Compare the recorded signal to the source signal with the following criteria:
    - Signal magnitude: Ensure that the source signal voltage magnitude is followed in the recorded signal. Any errors should be noted for evaluation and discussion at a later stage.
    - Frequency response: process and plot the frequency response of each signal (source and recorded) and compare the two, taking note of any discrepancies. Both the magnitude and phase of the frequency response should be evaluated.
    - Noise: Take note of any significant noise, observe any 'noise hot spot' frequency. When comparing each of the microphone boards' results at the end of the individual tests, take note of any trends in the noise corresponding to frequencies so that signal processing algorithms can be adjusted to filter out more noise at certain frequencies if needed.
- Finally, compare the live response of the microphone board to the source signal and take note of any initial delays, despite those to be expected at the set distance between microphone and source.

Every inaccuracy in the results of each board must be documented so that they can be accounted for when the microphones are tested simultaneously, and the final system is tested. Failure to document or adapt the system design to any inaccuracies in the individual tests could result in an entire system failure at a later stage.

6.2.2 Simultaneous Microphone Board Test

Simultaneous tests will be run on all 4 Adafruit I2S MEMS microphone breakout boards. Before these tests are run, both Raspberry Pi boards will be needed and used in this test, so it is assumed that they are both set up and programmed to be able to interact with the Adafruit I2S MEMS microphone breakout boards. The simultaneous tests are as follows:

- Place the sound source in the center of the grid and place all four microphones in a square, with the sound source in the center of the square. Each microphone should be placed the same distance from the sound source as they were placed in the first individual tests.
- Start recording through the microphone boards and then generate the same audio signal used in the individual tests above.
- Check that each microphone recorded the audio signal and that it matches the sound source projected.

6.3 Time delay estimation.

To test the time delay function of the system, the system shall be set up as described in the ***Simultaneous Microphone Board Test***, following which the tests below shall be run:

- Set the microphones to start recording data and use the sound source to generate an audio signal at a single frequency for as short a duration as possible.
- The system should record that the TDoA is 0.

Move the microphones to the outer corners of the grid and move the sound source to one corner next to one of the microphone boards.

- Run the TDoA function.
- Generate the same audio signal as above using the sound source.
- Verify that the values stored for the TDoA are non-zero, and that the TDoA value for the microphone close to the sound source is different to the other 3 values, which should be very similar.

6.4 Triangulation

Before testing the triangulation algorithm, it first should be verified that the distance calculation algorithm is working:

6.4.1 Distance Calculation Algorithm Test

- Place all four microphones in a square around the center of the grid (NOT on the outside corners of the grid, keep them close to the center), with the sound source directly in the middle of the square and run the distance calculation function.
- Emit a sharp burst of a single frequency from the sound source.
- Verify that the values calculated for the distances calculated are all equal.

- Move one microphone board to an outside corner of the grid and repeat the first 2 steps above. This time the distances calculated should be different.

Once the distance calculation algorithm has been verified, the triangulation algorithm can be tested.

6.4.2 Triangulation Algorithm Test

- Repeat the steps as shown in the ***Distance Calculation Algorithm Test***, running the triangulation function instead of the distance function.
- Verify that the triangulation function can accurately locate the sound source in the center of the grid for both tests.

Further Triangulation Tests:

- Move the microphones to the corners of the grid and move the sound source by at least 20 cm in any direction.
- Run the triangulation function and emit a sharp, single frequency audio signal from the sound source.
- Verify that the function correctly locates the sound source.
- Move the sound source by smaller distances (1cm or less movements) and verify that the accuracy of the triangulation function is as expected.

6.5 Varying frequency audio signal

Run the final set of tests in the ***Triangulation Algorithm Test***, but with a varying frequency audio signal instead of a single frequency and verify that the system can still locate the sound source for all test cases.

6.6 Noise robustness

Single Frequency:

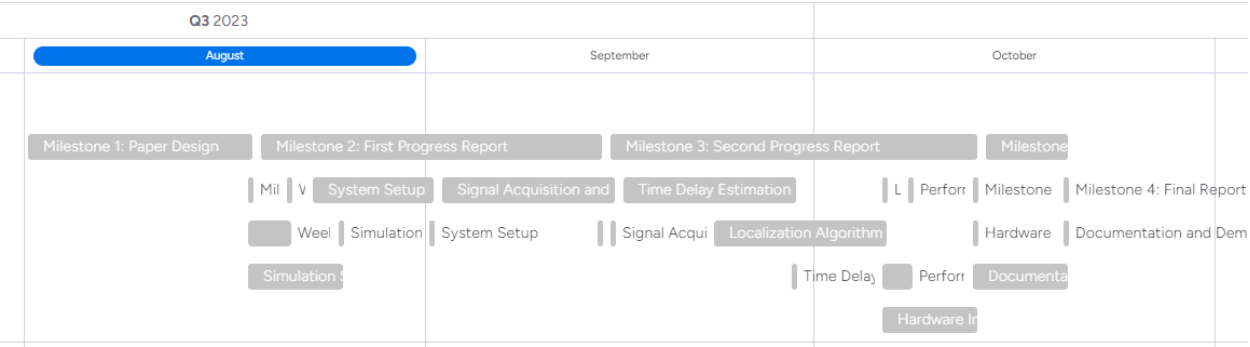
Run the ***Triangulation Algorithm Test*** steps above, but add in simulated noise such as a fan running in the background, music playing on a speaker, talking quietly etc. Start with a single source of quite noise and gradually increase the noise levels, running the triangulation test each time. If the system accurately manages to locate the sound source each time, continue with the following tests below.

Varying Frequency:

Run the ***Varying frequency audio signal*** tests above, incorporating the noise addition as described in the section above for noise tests with a single frequency.

7 Project Development Timeline

Please note that the visibility is quite poor as the website did not provide an option for better viewing, please see link below for clarity.



https://myuct895568.monday.com/boards/1251752474/views/5898951

8 Bibliography

Boschen, D. (2018, October 4). *GPD CA Signal Aquisition*. Retrieved from StackExchange: https://dsp.stackexchange.com/questions/52370/gps-ca-signal-acquisition

Fernandez, A., & Dang, D. (2013). Chapter 8 - Analog: The Infinite Shades of Gray. In A. Fernandez, & D. Dang, *Getting Started with the MSP430 Launchpad* (pp. 81-125). Retrieved from https://www.sciencedirect.com/science/article/abs/pii/B978012411588000008X

Fromaget, P. (2019). *How to Sync Time with a Server on Raspberry Pi*. Retrieved from RaspberryTips: https://raspberrytips.com/time-sync-raspberry-pi/

INTERSPEECH2021. (2022, January 10). *Time Delay Estimation for Speaker Localization Using CNN-Based Parametrized GCC-PHAT Features - (Oral presentation)*. Retrieved August 16, 2023, from YouTube: youtube.com/watch?v=q4o9eGjE658

Khan, K. (2023, February 1). *Time Synchonization on the Raspberry Pi*. Retrieved from Hackster.io: https://www.hackster.io/kamaluddinkhan/time-synchronization-on-the-raspberry-pi-d11ece

MATLAB. (2014, December 4). *Determining Signal Similarities*. Retrieved August 16, 2023, from YouTube: https://www.youtube.com/watch?v=oCcUm0_rUJw

Mohan, J., Pandu, V., & Kanagasabai, A. (2019). Real-Time ECG-Based Biometric Authentication System. In S. Geetha, & A. V. Phamila, *Countering Cyber Attacks and Preserving the Integrity and Availability of Critical Systems* (pp. 275-289). doi:10.4018/978-1-5225-8241-0

NI. (2023, February 21). *LabVIEW Sound and Vibration Toolkit*. Retrieved from ni: https://www.ni.com/docs/en-US/bundle/labview-sound-and-vibration-toolkit/page/svtconcepts/svsyncsampling.html

Rajashekar, U., & Simoncelli, E. P. (2009). Chapter 11 - Multiscale Denoising of Photographic Images. In U. Rajashekar, & E. P. Simoncelli, *The Essential Guide to Image Processing* (2nd ed., pp. 241-261). Retrieved from https://www.sciencedirect.com/science/article/abs/pii/B9780123744579000111

TechOverflow. (n.d.). *How to enable or disable NTP time synchronization on the Raspberry Pi*. Retrieved from TechOverflow: https://techoverflow.net/2023/03/14/how-to-enable-or-disable-ntp-time-synchronization-on-the-raspberry-pi/

Tips, R. (2020, March 5). *How to sync time with a server on Raspberry Pi?* Retrieved August 17, 2023, from YouTube: https://www.youtube.com/watch?v=Nyr5DI0fuAY

Weck, P. O. (2015). *Fundamentals of Systems Engineering.* Cambridge: Massachusetts Institute of Technology.

Zamir, Z. (2023, March). *How to Sync Time with a Server on Raspberry Pi*. Retrieved from linuxhint: https://linuxhint.com/sync-time-with-a-server-on-raspberry-pi/

References

1. YouTube Video:

Title: Raspberry Pi Time Synchronization Tutorial

URL: How to sync time with a server on Raspberry Pi?

2. Blog Post:

Title: How to Sync Time on Raspberry Pi (Step by Step Guide)

URL: https://raspberrytips.com/time-sync-raspberry-pi/

3. Hackster Project:

Title: Time Synchronization on the Raspberry Pi

URL: https://www.hackster.io/kamaluddinkhan/time-synchronization-on-the-raspberry-pi-d11ece

4. LinuxHint Article:

Title: How to Sync Time with a Server on Raspberry Pi

URL: https://linuxhint.com/sync-time-with-a-server-on-raspberry-pi/

5. TechOverflow Article:

Title: How to Enable or Disable NTP Time Synchronization on the Raspberry Pi

URL: https://techoverflow.net/2023/03/14/how-to-enable-or-disable-ntp-time-synchronization-on-the-raspberry-pi/

6. IGI Global Dictionary Entry:

Title: Signal Acquisition

URL: https://www.igi-global.com/dictionary/signal-acquisition/70639

7. ScienceDirect Topic:

Title: Signal Acquisition

URL: https://www.sciencedirect.com/topics/engineering/signal-acquisition

8. National Instruments Documentation:

Title: Sync Sampling

URL: https://www.ni.com/docs/en-US/bundle/labview-sound-and-vibration-toolkit/page/svtconcepts/svsyncsampling.html

9. DSP Stack Exchange Post:

Title: GPS CA Signal Acquisition

URL: https://dsp.stackexchange.com/questions/52370/gps-ca-signal-acquisition

10. Google Search Results:

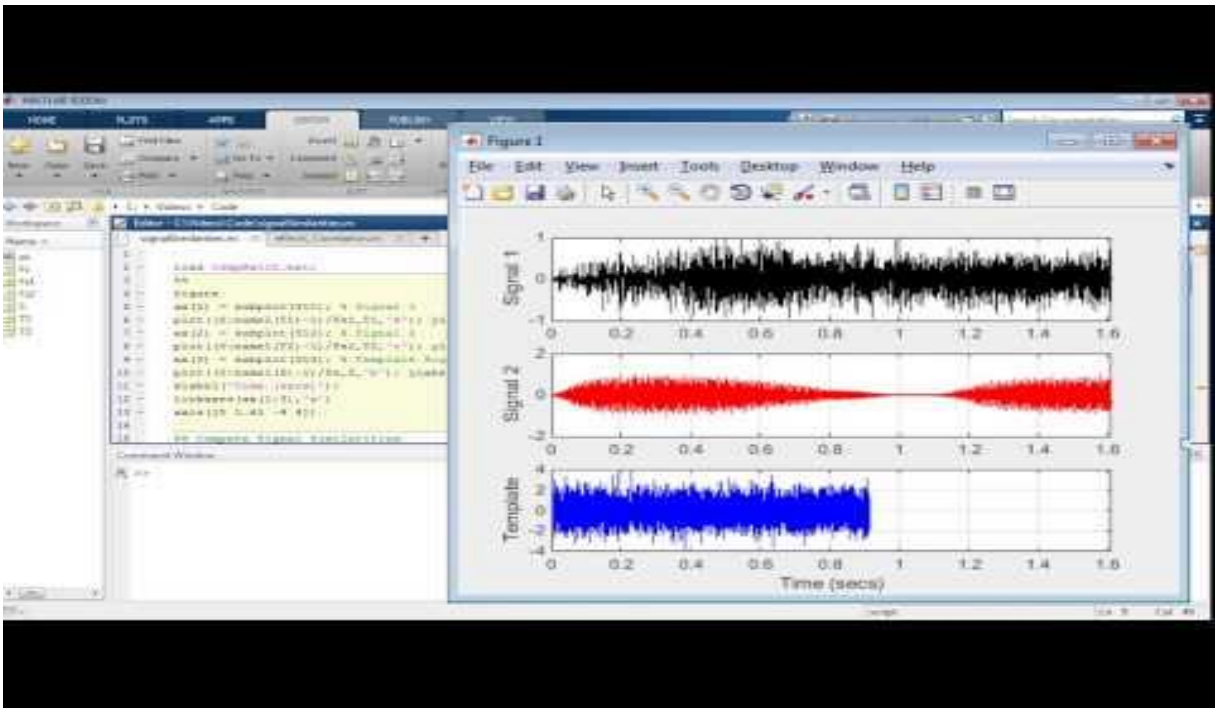
Title: Signal Acquisition Asynchronous Sampling with Cross-Correlation

URL: https://www.google.com/search?sca_esv=557612269&rlz=1C1CHBF_enZA1064ZA1064&q=Signal+Acquisition+Asynchronous+sampling+with+cross-correlation.&tbm=vid&source=lnms&sa=X&ved=2ahUKEwje3oDer-KAAxVhVeUKHfOTAmwQ0pQJegQIDBAB&biw=1280&bih=606&dpr=1.5#fpstate=ive&vld=cid:04b9b12b,vid:KkM5tZ_hvoY

11. YouTube Video:

Title: Cross-Correlation and Its Applications

URL: Determining Signal Similarities



12. YouTube Video:

Title: Introduction to Cross-Correlation in Digital Signal Processing

URL: [Time Delay Estimation for Speaker Localization Using CNN-Based Parametrized GCC-PHAT Features - ...]

13. YouTube Video:

    Title: Cross-Correlation Function in DSP

    URL: https://www.youtube.com/watch?v=dPjlmtGGMoM

14. Engineering Reports Article:

    Title: Real-Time Signal Acquisition and Processing for Localization

    URL: https://onlinelibrary.wiley.com/doi/full/10.1002/eng2.12236

15. ScienceDirect Article:

    Title: Signal Acquisition Using Dual-Antenna GPS

    URL: https://www.sciencedirect.com/science/article/abs/pii/S0263224119306359

16. arXiv Preprint:

    Title: Generalized Cross-Correlation for Time-Delay Estimation

    URL: https://arxiv.org/abs/1910.08838

17. MDPI Article:

    Title: Generalized Cross-Correlation Algorithm for Time Delay Estimation of Multi-Channel Signals

    URL: https://www.mdpi.com/1424-8220/20/18/5190

18. IEEE Xplore Article:

    Title: Fast Normalized Cross-Correlation

    URL: https://ieeexplore.ieee.org/document/1040634

19. EURASIP Journal Article:

Title: Improved Triangle Area-Based Signal Source Location Algorithm in Complex Environments

URL: https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-022-02109-3

20. Scientific Research Publishing Article:

Title: Improved Signal Acquisition Method Based on the Cross-Correlation Function in GPS Receiver

URL: https://www.scirp.org/journal/paperinformation.aspx?paperid=74255

21. PathPartner Tech Article:

Title: Triangulation vs. Trilateration vs. Multilateration for Indoor Positioning Systems

URL: https://www.pathpartnertech.com/triangulation-vs-trilateration-vs-multilateration-for-indoor-positioning-systems/

22. Taylor & Francis Article:

Title: Signal Acquisition and Analysis

URL: https://www.tandfonline.com/doi/abs/10.1179/sre.1950.10.78.353

23. Python GUIs Comparison:

Title: PyQt vs Tkinter - Which GUI Framework is Better?

URL: https://www.pythonguis.com/faq/pyqt-vs-tkinter/#:~:text=If%20you%20are%20learning%20Python,sense%20to%20start%20with%20PyQt.

24. Semantic UI Official Website:

URL: https://semantic-ui.com/

25. Geekflare Article:

Title: 10 Best JavaScript UI Libraries for Developers

URL: https://geekflare.com/best-javascript-ui-libraries/

26. Red Hat Developer Article:

Title: Ticket Monster User Frontend

URL: https://developers.redhat.com/ticket-monster/userfrontend

9 Appendix

9.1 Term definitions

9.1.1 Shared Network Time Protocols (NTP)

Utilizing network time protocols (NTP) to synchronize multiple devices, ensuring consistent time across a network.

9.1.2 Hardware-Based Synchronization

Achieving synchronization by using specialized hardware components, such as GPS receivers or atomic clocks, to align device time.

9.1.3 Software-Based Synchronization

Employing software-generated clock signals to synchronize devices, aligning their local clocks for coordinated timekeeping.

9.1.4 Standalone Graphical Application using:

Creating an independent visual program utilizing libraries like PyQt or Tkinter for constructing the user interface and managing interactions.

9.1.5 Integration of Web-Based Interface with HTML, CSS, and JavaScript Frameworks:

Combining a browser-based interface by employing HTML, CSS, and JavaScript frameworks, enabling dynamic and visually appealing interactions over the web.