3. Project 1. Acoustic Link

Important

Please read the following instructions carefully:

1. This project is to be completed by each group individually. Use your own code to complete the main parts of the tasks.
2. This project contains 8 tasks. 5 of them are optional (10 points + 8 points).
3. The task score does not reflect the implementation difficulty but rather the recommended level.
4. Suggested workload is 4~6 FULL days. Manage your time.
5. Submit your code through Blackboard. The submission due date is **Oct. 12, 2025**.
6. Each group needs to submit the code once and only once. Immediately after TAs' checking. The submission is performed by one of the group members.
7. Tasks with "Optional" tag are optional tasks. Their due date is the end of the semester. Optional tasks are graded based on performance criteria and the results of the code review.
8. Unless otherwise mentioned, the instructor is responsible for grading optional tasks. Contact the instructor to check if you have finished one or more of them.
9. *Task4* and *Task 8* are graded by TAs.
10. Tasks are graded according to their hierarchy. The hierarchy of this project is *Task 1* < *Task 2* < *Task 3*. A full score of one task automatically guarantees the full score of non-overdue preceding tasks (left side of "<").
11. During the performance assessment, any task can only be attempted up to 5 times.

3.1. Overview

Project 0 introduces the basic usage of audio interfaces. This project will build a (not necessarily 100% reliable) data link between two air-gapped computer `NODEs` by precisely manipulating and sensing sound signals. As shown in Figure 3.1, the tasks include defining and implementing modulation methods, frame structures, error correction mechanisms, and other essential elements of the physical-layer protocol. This layer also offers transmission (a.k.a., Tx) and reception (a.k.a., Rx) interfaces for upper layers.

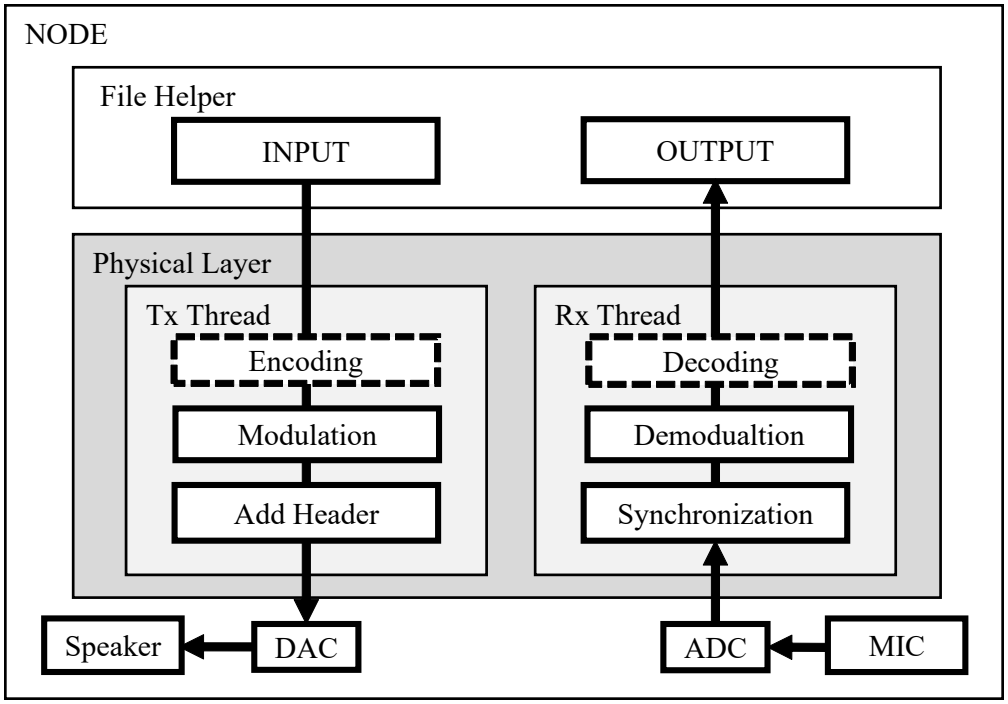

Figure 3.1. Project 1 Overview.

3.2. Task 0: (0 point) Audio Toolkit

Computers connect to audio peripherals through the audio connectors of the sound card. The typical use case involves inserting the 3.5 mm plug (M) of the microphone and/or speaker into the 3.5 mm jack (F) of the sound card. Due to the confusing structures of 3.5 mm audio connectors and diverse specifications of sound cards, we will provide the following toolkit. Please borrow it from TAs.

- USB Sound Card ×4

  This device includes 1 audio input connector (Red F), 1 audio output connector (Green F), and a USB type-A connector. When connected to the computer via USB, it will be recognized as a sound card. Its audio input/output can be read/generated, e.g., through ASIO. Note that while its connectors are all stereo, it only supports mono recording.

- Mixer ×1

  This device contains 4 input connectors (F) and 1 output connector (F). Its function is to combine the signals from 4 inputs and output them to the output connector. The strength of each input/output signal can be adjusted by knobs. All gain knobs must be turned fully maximized during performance assessment.

- Audio Cable ×6

  (M to M) audio cables, with lengths of 1.0 m or 0.5 m.

- 1-to-2 Cable Splitter ×3

This splitter has three internally-connected connectors (M, F, F). The input signal from any connector is split into the other two connectors, and the input signals from any two connectors are combined into a signal output to the remaining.

It is important to note that you can use 3.5 mm audio cables to connect computers only starting from Project 2. However, in Project 1, you can optionally use the sound card included in the kit. Additional cables can be requested for specific tasks, and replacements can be requested in case of suspected hardware issues.

3.3. Task 1: (3 points) Understanding Your Tools

Go back to Project 0. This task title is reserved here for task hierarchy. If this task is not submitted by the Project 0 deadline, this task will not be graded and counted.

3.4. Task 2: (2 points) Generating Sound Waves at Will

The sound card processes the numerical sequence $S = [s_1, \ldots, s_n]$ stored in its buffer, converting it into corresponding voltage levels at a fixed rate Fs to generate sound waves with corresponding power levels. S is known as audio samples, and Fs represents the audio sampling frequency. To generate an arbitrary sound signal $f(t)$, we calculate its samples periodically at a tick rate of fs, i.e., $S = [f(\frac{1}{\mathrm{fs}}), \ldots, f(\frac{n}{\mathrm{fs}})]$. By continuously appending S to the sound card's buffer, the played sound mirrors the waveform of $f(t)$ if and only if $\mathrm{fs} == \mathrm{Fs}$. If $\mathrm{fs} < \mathrm{Fs}$, it produces a faster version of the $f(t)$ sound, while $\mathrm{fs} > \mathrm{Fs}$ results in a slower version.

Therefore, to calculate S to generate $f(t)$, it is essential to set correct fs to Fs. Common Fs include 8 kHz, 16 kHz, 44.1 kHz, and 48 kHz. The higher the sampling frequency, the wider the frequency range S can represent. 48 kHz is the upper limit of mainstream sound cards because it already covers the human auditory frequency range. 48 kHz is also recommended for completing the course project. One reason is that it maximizes the utilization of the audio spectrum. Another reason is that when using other sampling frequencies, the sound card might implicitly perform resampling, introducing additional noise.

Tip

- You can record the sound into a file and then utilize tools like Matlab/NumPy to load, plot, and replay it. Debugging algorithms with offline files can be convenient as processes are reproducible.
- For real-time generation and visualization of sound signals, consider using smartphone apps like Tone Generator (for generating reference tones) and SpectrumView/Spectroid (for viewing waveforms and spectrum).
- An interesting fact is that the sampling rate of a sound card is determined by the clocks of its DAC/ADC. The tick rate of the clock might have a random offset from the claiming rate, e.g., 48 kHz. This offset is caused by various factors such as the clocking mechanism, voltage, and even temperature. So, it is not surprising to observe that the sound card actually consumes/produces several hundreds more/less samples per second.

Performance Assessment

The group provides one device: `NODE1`.

- Objective (2 points). `NODE1` should be able to play the signal: $f(t) = \sin(2\pi \cdot 1000 \cdot t) + \sin(2\pi \cdot 10000 \cdot t)$. TAs will use an acoustic spectrum analyzer to measure and verify the sharp frequency peaks at 1 kHz and 10 kHz.

3.5. Task 3: (5 points) Transmitting Your First Bit

If the signal $f(t)$ generated by the Tx node is audible at the Rx node, $f(t)$ can be used to transmit messages between the two nodes. The challenge lies in the fact that $f(t)$, after passing through the DAC, speaker, air gap, reaching the receiver's microphone, and undergoing ADC, becomes $f_{\text{rx}}(t)$, which is not precisely the same as the ideal $f(t)$. These differences include distortion, noise, and time shifts. Accurately and efficiently interpreting the information carried by $f_{\text{rx}}(t)$ relies on proper physical-layer designs.

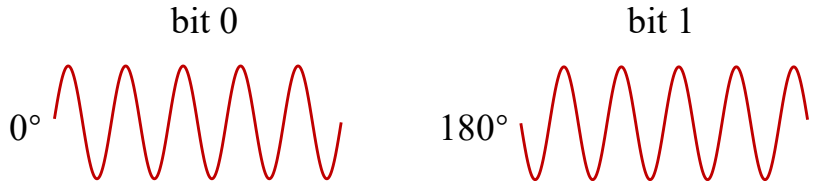

Figure 3.2. Example: Phase Shift Keying Modulation.

- Modulation and Demodulation.

  The Tx and Rx nodes must first agree on how to use $f(t)$ to represent information. One intuitive way is to map the bitstream into square waves, having corresponding binary amplitude variations at a certain rate. This approach is feasible when using cables as the transmission medium (starting from the next project). However, in the air propagation situation, simple practice can prove that this way may not be particularly effective. Due to the air propagation properties of sound waves as well as the device's engineering optimization for the human auditory range, not any Tx signals can be efficiently generated and perceived. Therefore, another choice is choosing an appropriate carrier sound wave to convey the bitstream.

  The amplitude, frequency, and phase of the carrier wave can be modulated according to the information to be transmitted. Demodulating it upon reception reveals the transmitted data. ASK (Amplitude Shift Keying), FSK (Frequency Shift Keying), and PSK (Phase Shift Keying) are three basic modulation methods that can be experimented with, along with any other modulation methods you prefer. Be aware that due to the instability of the sound amplitude, modulating the amplitude attribute might present unnecessary challenges.

  Figure 3.2 is an example of PSK modulation, where bits are mapped to symbols of different phase shifts, and the waveforms corresponding to the symbols are concatenated to form the signal $f(t)$ for transmission. Due to the (mechanical)

properties of microphones and loudspeakers, the transitions between symbols might last for a while. Thus, reserving adequate guard intervals between symbols helps mitigate inter-symbol interference.

Figure 3.3. Example: Physical Layer Frame Structure.

- Frame Structure.

  Organizing data into frames enables multiple nodes to share the physical channel effectively. The physical-layer frame structure, depicted in Figure 3.3, incorporates a frame header at the beginning, providing essential information to the receiver.

  The `Preamble`, a pre-defined signal, aids the receiver in determining whether a data frame is present in the detected sound. Note that when the received frame power closely matches the noise power, relying solely on energy detection for event occurrence is unreliable. The chirp signal is recommended as the preamble [sigcomm13]. Chirp signal is continuously frequency modulated. Even after propagation and distortion, it remains distinguishable from environmental noise and regular data signals. By correlating such preamble template with the received signal, valid frames can be detected based on the correlation output strength.

  Typically, the `Preamble` field also serves to synchronize the clocks of the Tx and Rx nodes. Chirp signals exhibit excellent autocorrelation properties. A significant local peak in the correlation amplitude occurs when it aligns with the preamble of the received frame. This method achieves precise synchronization accuracy (at the sample level), especially crucial at high symbol rates. Utilizing the synchronized sample position, the receiver can ascertain symbol boundaries, facilitating demodulation with reduced inter-symbol interference.

  During the demodulation process, the receiver needs to know when this process ends. Several approaches handle this. In the Example frame, a `Length` field describes the number of symbols within the frame. The receiver can dynamically obtain this value shortly after detecting the Preamble.

  Additionally, we recommend incorporating a `CRC` (Cyclic Redundancy Check) field to validate the accuracy of received information. Although this task does not demand 100% reliable transmission, error awareness prevents delivering incorrect frame payloads to upper layers. Such errors could lead to more severe issues than the errors themselves, such as incorrect control messages.

Tip

- Please refer carefully to the physical layer implementation `SamplePHY.m` on Blackboard.

- Multiple carriers can be used simultaneously to increase throughput. Ensure ample frequency intervals between carriers to prevent frequency domain interference.
- Due to the subtle offsets in sound cards' sampling frequencies, the synchronized sampling point obtained using frame header still gradually drifts. For excessively long frames (although we do not recommend this design), periodic resynchronization becomes necessary during demodulation.
- Avoid short frame headers. Loudspeakers and microphones require time to warm up.
- Use balanced signals, i.e., the number of positive and negative sample points is approximately equal and they are almost interwoven with each other, to drive the loudspeakers for optimal performance.
- If you notice occasional buffer underruns in the sound card, causing the loss of some samples during playing or recording, it might be due to delayed scheduling of the sound card driver. To enhance the real-time performance, switch your computer's power management to the `performance mode` to disable CPU power-saving features [atc21].
- Successive volume might lead to more noise.
- The quality of microphone and loudspeaker can significantly affect the performance.

Performance Assessment

The group provides two devices: `NODE1` and `NODE2`

- Objective (5 points). TAs provide `INPUT.txt`, which contains 10,000 "0"s or "1"s. `NODE1` sends the bits from the file to `NODE2`. `NODE2` stores the received bits in `OUTPUT.txt`. During the transmission, TAs keep silent.

  Transmission must be completed within 15 seconds:

Completion Time	Percentage Earned
<15 s	100%
>15 s	0%

  TAs use `diff` tool to compare `INPUT.txt` and `OUTPUT.txt`:

Similarity	Percentage Earned
<80%	0%
<100%	80%
100%	100%

3.6. Task 4: (Optional, 1 point) Error Correction

"过而能改，善莫大焉" —《左传》

Errors are nearly inevitable in network transmissions. When the Bit Error Rate (BER) is high, retransmission is not very efficient because a high BER, like 1/100, could lead to almost all frames, including the retransmitted ones, containing errors. Forward Error Correction (FEC) codes provide error correction capabilities to the receiver by adding redundancy to the original data and encoding them, thus minimizing the need for retransmission. RS codes, convolutional codes, LDPC codes, Polar codes, and others have efficient implementations and wide applications. Implement an FEC code to enhance the performance of the acoustic link.

Performance Assessment

Similarity of `INPUT.txt` and `OUTPUT.txt`:

Similarity	Percentage Earned
<100%	0%
100%	100%

Other assessment criteria and procedures are the same as in Task 3, along with **code review**.

3.7. Task 5: (Optional, 2 points) OFDM

When attempting to further increase the bandwidth of the audio channel, one approach is to raise the baud rate, i.e., reducing the symbol duration. However, several factors impose limits on symbol duration. As mentioned earlier, due to constraints in transducer components, instantaneous switching between symbols is not possible. Adequate transition intervals need to be reserved. Additionally, the slow propagation speed of sound introduces obvious interference between symbols due to multipath copies (similar to echoes). Another choice is to increase the number of simultaneous carriers. However, as stated earlier, interference can occur among nearby carriers.

It is worth noting that these issues are not exclusive to the audio channel. Orthogonal Frequency Division Multiplexing (OFDM), a popular design in modern communication systems, provides solutions for these challenges. OFDM can extend symbol duration while simultaneously utilizing multiple carriers to maintain the transmission rate. Its key feature lies in isolating interference among carriers, achieving highly efficient spectrum utilization. So, please implement OFDM to finish this task.

Performance Assessment

The assessment criteria and procedures are the same as in Task 4.

3.8. Task 6: (Optional, 2 points) Chip Dream

Almost all commercial network card's physical layer processing is done through circuits. Software-defined radio was attempted in the past [CACM11] but was never adopted by consumer products. Please consider optimizing your implementation from a hardware perspective. In hardware circuits, floating-point operators are expensive, so please use

integers or fixed-point numbers to improve the implementation. Avoid using multiplication, division, and complex functions like sine and cosine functions.

Tip

- Use look up table to implement complex functions.
- Some ASIO wrappers expose data interface in float format. To complete this task, please do an additional conversion from float/double to `INT32`. Then, just pretend to ignore the conversion.
- This task may be incompatible with other optional tasks or require an excessive workload.

Performance Assessment

The assessment criteria and procedures are the same as in Task 4.

3.9. Task 7: (Optional, 1 point) MIMO

Multiple Input Multiple Output (MIMO) systems [CCR10] are a significant advancement in modern communication systems. Radio MIMO systems employ multiple antennas to simultaneously transfer multiple data streams within the same frequency band. The receiver distinguishes these *overlapping* streams through leveraging the differences, i.e., phase shifts and amplitude attenuation, in the propagation paths between different Tx and Rx antennas pairs. These differences are formally described in CSI (channel state information) and can be measured using predefined patterns located after the synchronization preamble and before valid data symbols.

For this task, please prepare two microphones for the Rx node and two speakers for the Tx node, and refer to radio MIMO designs, e.g., a neat example from WARP project, to implement a 2×2 audio MIMO system. MIMO systems rely on precise synchronization to separate concurrent streams, so they usually use the same clock to sample signals received in different Rx paths. You can reach out to the TA to borrow the audio tool set exclusively for this task:

- USB audio card supporting stereo recording ×1
- Microphone with XLR output ×2
- Loudspeakers are not included

Tip

- While MIMO is often used with OFDM in modern communication systems, it is not mandatory. Implementing MIMO with a single carrier is more straightforward.

Performance Assessment

`NODE1`'s speakers are positioned at least 40 cm away from `NODE2`'s microphones.

Transmission completion time:

Completion Time	Percentage Earned
<20 s	100%
>20 s	0%

Other assessment criteria and procedures are the same as in Task 4.

3.10. Task 8: (Optional, 2 points) Range Challenge

After propagating over longer distances, sound waves exhibit richer and more complex characteristics, such as multipath effects, making data transmission more challenging. This task is aimed at encouraging groups achieving longer transmission distances.

Performance Assessment

Separation distance of `NODE1`'s speaker and `NODE2`'s microphone:

Distance	Percentage Earned
>200 cm	100%
>150 cm	75%
>100 cm	50%
>50 cm	25%

Other assessment criteria and procedures are the same as in Task 4.

References

[atc21] Understanding precision time protocol in today's wi-fi networks. https://www.usenix.org/system/files/atc21-chen.pdf

[sigcomm13] Dhwani: secure peer-to-peer acoustic NFC. https://doi.org/10.1145/2486001.2486037

[CCR10] 802.11 with Multiple Antennas for Dummies https://doi.org/10.1145/1672308.1672313

[CACM11] Sora: high-performance software radio using general-purpose multi-core processors https://doi.org/10.1145/1866739.1866760