Name: \_\_Alex Koo, Jason Ikonomopoulos\_\_\_\_

EID: \_\_gk7244, pji96\_\_\_\_

Semester: Fall 2024

Course: ECE445L

1. ***Objectives*:**

1. In a few sentences, describe the purpose of the lab and the features of your system.

The overall purpose of this lab is to get introudction to the field of audio signal processing. With smaller goals of studying ADC conversion and Nyquist theroem, characterizing the perofrmance of the DAC and ADC, encoding information as sound output from the DAC to the speaker, decoding information as sound input from the microphone to ADC, and developing an audio communication System.

2. ***Hardware Design Deliverables:***

1. Deliverable 1: Using KiCad, create a schematic or figure showing all external components connected to the TM4C123 board. You do not need to show hardware components on the TM4C123 LaunchPad board. Include a screenshot below.

A diagram of a circuit board

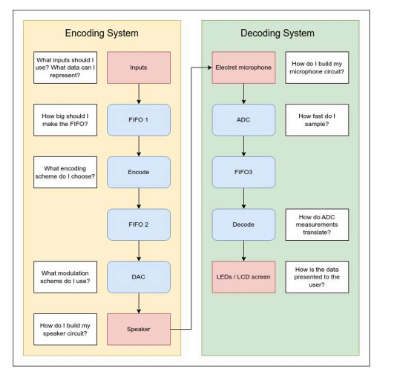
Description automatically generated

3. ***Software Design Deliverables:***

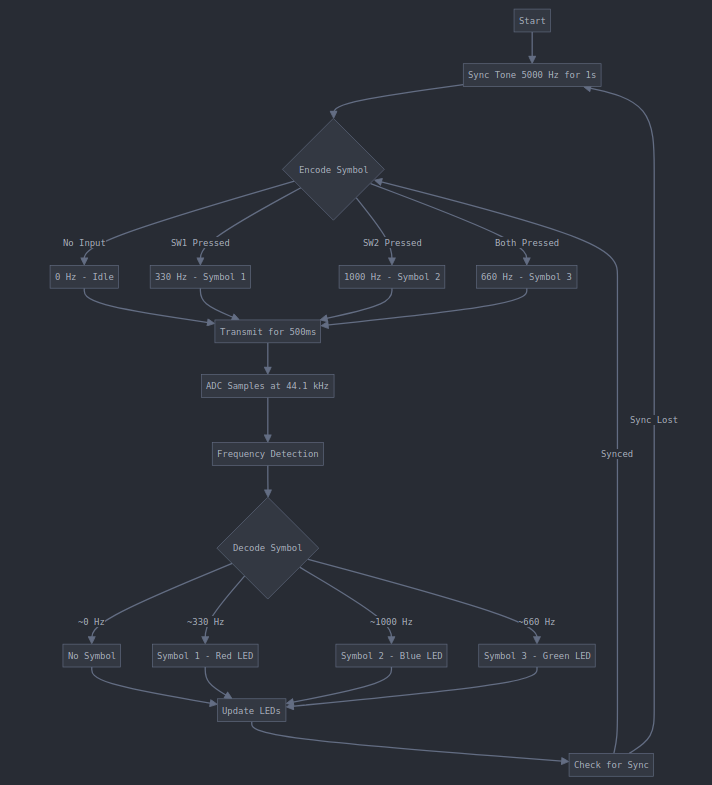
I have pushed my code to GitHub for grading (Write yes if true): yes

1. Briefly describe the system design. Include a data flow and call graph if your system is different than Figure 5.1 and 5.2.

We kept our system design the same as the one proposed in the lab document that is below.



1. Describe your message encoding scheme. Include any diagrams or graphs.



Before transmitting data, a sync tone of 5000 Hz is sent for 10 milli seconds (SYNC\_DURATION\_MS = 1000). This sync tone helps the receiver synchronize and prepare for incoming data. Symbol Encoding. The scheme uses four distinct frequencies to represent different symbols:

* + FREQ\_SYMBOL0 (0 Hz): No sound, represents no button pressed or idle state
  + FREQ\_SYMBOL1 (330 Hz): Represents Symbol 1, typically when SW1 is pressed
  + FREQ\_SYMBOL2 (1000 Hz): Represents Symbol 2, typically when SW2 is pressed
  + FREQ\_SYMBOL3 (660 Hz): Represents Symbol 3, typically when both buttons are pressed
* Each symbol is transmitted for a fixed duration of 500 ms (OUTPUT\_DURATION\_MS = 500). The DAC (Digital-to-Analog Converter) generates a sine wave at the corresponding frequency for each symbol.
* The receiver samples the incoming audio signal at 44.1 kHz (SAMPLE\_RATE = 44100). It uses a simple zero-crossing detection method to estimate the frequency of the received signal. The detected frequency is then matched to the closest symbol frequency, with a threshold of 50 Hz (FREQ\_THRESHOLD = 50) for noise tolerance.

4. ***Measurement Data:***

1. Deliverable 2: Quantized waveform of DAC output dump and relevant context.

A screen with a graph on it

Description automatically generated

1 kHz DAC output

A screen with a graph on it

Description automatically generated

2 kHz DAC output

1. Deliverable 3: Quantized waveform of ADC input dump and relevant context.

A screen with a screen showing a wave

Description automatically generated

330 Hz signal played into the microphone

A screen with a screen showing a wave

Description automatically generated

660 Hz signal played into the microphone

A screen with a screen showing a sound wave

Description automatically generated with medium confidence

1 kHz signal played into the microphone

1. Deliverable 4: System module encoder and decoder CPU utilization.

The system module encoder and decoder CPU utilization are somewhat close to each other in value with decoder being slightly more due to zero crossing processing being used to decode the incoming signals. The encoder CPU usage was primiarily driven by Timer0A interrupts though encoding itself adds negligible CPU overhead. The main CPU intensive tasks was on the decoding as it processed large blocks of samples (1024 at a time), the frequency detection algorithm processed every sampel in the block, and the processing happens in the main loop.

1. Deliverable 5: Quantification of system performance.

Microphone frequency detection error:

Test Frequency 1: 330 Hz

Measured Frequency 1: 328 Hz

Frequency 1 Error: (330 – 328) / 330 = 0.6% error

Test Frequency 2: 1000 Hz

Measured Frequency 2: 997 Hz

Frequency 2 Error: (1000 - 997) / 1000 = 0.3% error

Test Frequency 3: 5000 Hz

Measured Frequency 3: 4995 Hz

Frequency 3 Error: (5000 – 4995) / 5000 = 0.1% error

5. ***Analysis and Discussion Questions:***

1. What is the Nyquist Theorem and Valvano Postulate and how do they apply to this lab?

The TM4C maximum sampling rate is 125k samples/sec. Since the highest frequency component in the audio signals in this lab is well below the microcontroller’s maximum sampling capability, there is a comfortable margin to satisfy the Nyquist Theorem. Following the Valvano postulate ensures an even higher fidelity of the reconstructed audio signal because it goes beyond the minimnum Nyquist rate. By sampling at more than 10 times the highest audio frequency component, the digitral samples will more accurately represent the original signal, leading to clearer and more precise processing and transmission of audio data in your system.

1. How did you eliminate noise in the sampled audio?

We struggled with noise reudciton in our system but the main way we tried to eliminate it was by choosing an apporpitate sampl;ing rate according to the Valvano postulate, we applied oversampling techniques. We also attempted on using FFT algorithm to reduce noise and averaging to reduce random noise.

1. How does your protocol allow (or doesn’t allow) communication in the presence of background noise?

Our protocol employs a frequency-based encoding scheme that offers some inherent resistance to background noise. We utilize distinct frequencies (0 Hz, 330 Hz, 660 Hz, and 1000 Hz) for symbol representation, which helps distinguish our signal from broadband noise. A key feature of our system is the 5000 Hz sync tone transmitted for 1 second before data transmission, allowing the receiver to recognize transmission start and potentially re-establish communication if it's temporarily lost due to noise. Our frequency detection method uses zero-crossing detection on a large buffer of 1024 samples, which can help average out some noise effects and improve detection accuracy. We've implemented a frequency threshold of 50 Hz for symbol matching, providing some tolerance for frequency deviation in noisy conditions. Our system continuously transmits symbols, either data or an idle tone, which aids in maintaining synchronization in noisy environments. The fixed symbol duration of 500 ms gives the receiver ample time to detect the correct frequency, further enhancing noise resistance. However, our protocol has some limitations in highly noisy environments. The simple zero-crossing detection method might be susceptible to noise that causes additional zero crossings, and we lack error correction mechanisms beyond basic frequency matching. We also don't have adaptive mechanisms to adjust to changing noise conditions. Despite these limitations, our protocol's use of distinct frequencies, sync tones, large sample sizes for detection, and frequency thresholds provides a foundation for communication in the presence of moderate background noise.

1. How could you improve the bandwidth or baud rate of your system?

Increase symbol rate: We could reduce the current 500ms symbol duration, allowing for faster transmission of symbols. However, this would need to be balanced against the need for accurate frequency detection.

Use more frequencies: By expanding our frequency set beyond the current four (0 Hz, 330 Hz, 660 Hz, 1000 Hz), we could encode more information per symbol. For instance, using 8 or 16 distinct frequencies would allow us to transmit 3 or 4 bits per symbol instead of 2.

Implement more efficient coding: Use data compression or more efficient encoding schemes to transmit more information with fewer symbols.

Parallel transmission: If hardware allows, we could transmit on multiple channels simultaneously.