THE UNIVERSITY OF TEXAS AT ARLINGTON

COMPUTER SCIENCE AND ENGINEERING

EMBEDDED II TERM PROJECT REPORT

EMBEDDED SYSTEM II

Submitted toward the partial completion of the requirements for CSE 4342-001

Submitted by,

Servando Olvera

1001909287

Date 05/02/2024

Approach For Design

The goal of the project was to design a low-cost, low-power device capable of solving the angle of arrival problem in continuous audio. This audio was to be picked up by three microphones configured as analog inputs and processed by Sample Sequencer 1 (SS1) on the Analog to Digital Converter 0 (ADC0) peripheral. SS1 was chosen because it could store up to four samples, one from each mic, on its FIFO. Subsequently, since continuous audio entails the need for multiple samples to be collected, the Micro Direct Memory Access (DMA) peripheral was used as an audio acquisition system in conjunction with ADC0. The idea was that for every sample collected on the FIFO, the uDMA would grab it and store it in a much larger array that could later be accessed if desired. DMA was chosen for its efficiency and relative ease of operation. Once this peripheral was set and configured, it would run automatically in the background, constantly storing data from FIFO. The configuration used for the uDMA was that of Ping-Pong: a configuration that entails the use of two buffers being filled up one right after the other. However, in hindsight, the Ping-Pong configuration was not entirely necessary since one buffer would have sufficed. Furthermore, a sound detection feature was implemented using digital comparators from the ADC1 peripheral (The reason ADC0's digital comparators were not used is that these peripherals cannot use the FIFO and the digital comparator simultaneously). This feature would allow the device to determine a valid event to be processed in a noisy or quiet environment. The digital comparators were configured as high band, with the threshold being dynamic; constantly changing every one-fourth of a second, as the noise in the environment changes. The reason for using digital comparators over analog was due to faster processing of audio and much higher precision. On top of this, the ADC1 peripherals allowed for data collected by it to be compared instantly against a threshold value on these digital comparators, all of which were linked to a respective microphone, so if one were to detect a rise in volume, a valid event would be triggered. Now, once a valid event was triggered, the idea was to allow the uDMA to fill up the current buffer and then the other one. This would ensure some past data to be kept and some future data to be collected right after the trigger had been set. Upon filling up the second buffer, everything would be stopped, and the data would be processed. First, the data was separated into three arrays for each respective microphone, and a cross-correlation was performed between the three afterward. From the cross-correlation, the order in which the mics got hit and the time difference between all mics was obtained. Having this information, the reference angle (0˚, 120˚, 240˚) could be determined, which was the first mic to get hit. Based on the second mic to get hit, it would entail either adding or subtracting degrees, and the time difference of the second and third mic to get hit would be multiplied by a constant of 1.05. This number would then be added to the reference angle to obtain the angle of arrival, and thus complete the approach to the problem.

Peripherals Used and Their Purpose

General Purpose Inputs/Outputs (GPIO)

This peripheral was used to associate the outputs of the mics with a pin in the microcontroller. Four pins were configured as analog inputs, each belonging to one of the respective mics to have access to its value in software. Additionally, as extra credit three pins from port F were configured to use the Red, Blue and Green LEDs.

Analog to Digital Converter (ADC)

ADC1/0 was used to convert continuous analog voltage from the mics to a discrete digital number. The purpose of this was so that the output of the mics could be interpreted in software in a meaningful manner. ADC0 was used in conjunction with the uDAM so that each sample from each mic was being stored right away by the uDMA. ADC1 was used as a trigger, so that if the surrounding sound went above a threshold, the data would begin to be processed.

Micro Direct Memory Access (uDMA)

The purpose of the DMA was to store away the data from ADC0 SS1 as it was being collected into the FIFO, so that once an event was triggered enough data would be there to be processed.

General Purpose Timers (GPTM)

Timer 1a was configured as a periodic timer that would go off every one-forth of a second to calculate the new threshold for the digital comparators. Time 2a was configured as a one-shot timer that would be the holdoff time before a new event could be processed.

Universal Asynchronous Receiver/Transmitter (UART)

UART was used to be able to communicate with the device. Three commands were implemented so that there would be a level of control by the user. Command ‘aoa’ would display the latest angle of arrival. Command ‘aoa always’ would display the angle of arrival as an event occurs. Command ‘tdoa’ would display the time difference of angle.

Pulse Width Modulation (PWM)

This peripheral was used for extra credit to map the angle of arrival to a respective RGB value. This value was represented according to an intensity from the three LEDs, an intensity which was set using PWM.

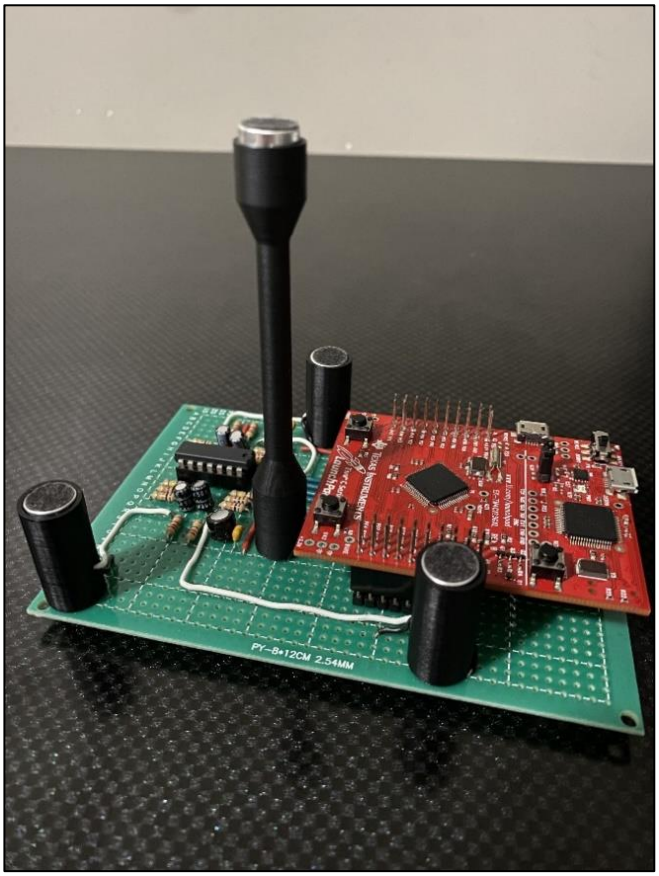
Algorithm and Logic

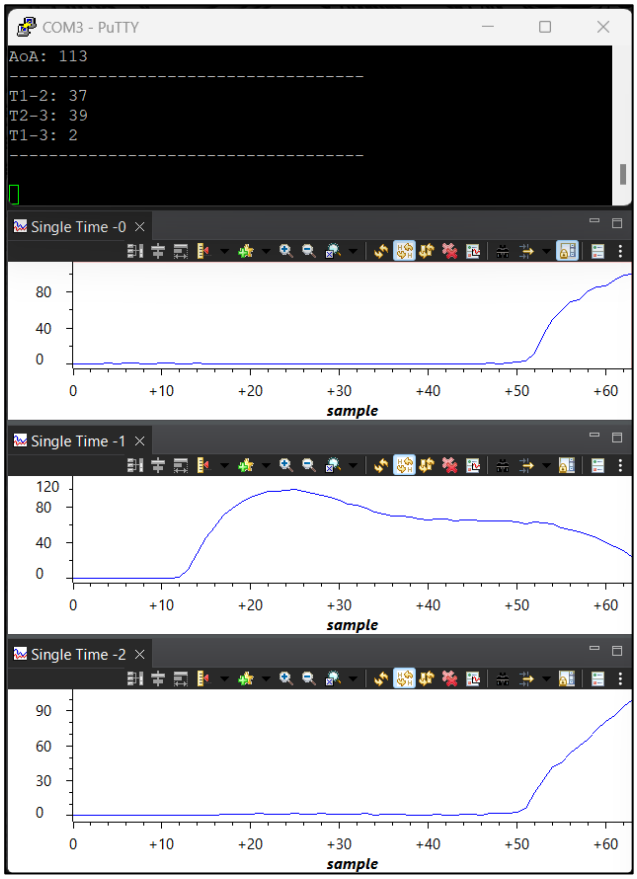
Once the data acquisition system had been configured, and the triggers had been armed for the validation of an event, all the was left to do was implement the logic necessary to process the data and calculate the angle of arrival. The idea that I implemented basically entailed that upon the detection of a valid event, basically sound in the room going above a set threshold, a flag would be set to indicate that an event has begun and the ADC1 would be disabled to prevent consecutive triggers after the initial one. Then, once the uDMA entered it periodic interrupt with this flag set it would indicate that one out of the two buffers had gotten full, so the flag would be updated to indicate just that. The uDMA would then be allowed to run another instance to fill the second buffer and upon entering the interrupt again, the flag would be updated it to indicate that the data can now be processed and the uDMA would be disabled along with ADC0. One to note is that a global variable was used to keep track of which buffer got full last. Down in main, the previously mentioned flag is constantly being checked and upon changing to "PROCESS_DATA" the following was performed: the data on the uDMA was separated, since it contains mixed data from all mics, into four separate buffers. This data was then normalized by multiplying each item in each array by one hundred and then dividing it by the highest value in that respective array. This would give me the same scale on the date for each mic. Having the data normalized, cross-correlation was performed between MIC1 and MIC2, MIC2 and MIC3, and MIC1 and MIC3. The cross correlation would determine the shift necessary for the data in mics correlated to match with each other. One thing to note is that some minor adjustments were made to the results of the cross-correlation since the shift wasn't always accurate. Having these shifting values, it was then possible to calculate the angle of arrival and so a simple algorithm was implemented to determined which mic got hit first and thus obtain θ_O (0˚, 120˚, 240˚). The logic entailed something like this: if MIC1 had to shift forward (positive shift) to match with MIC2 and it also had to shift forward to match with MIC3, then MIC1 got hit first, and θ_O would be 0˚. Then, the second mic to get hit would be determined, with the same logic just described, to either add or subtract from θ_O. The degrees to be added or subtracted were dependent on the absolute time different between the second and third mics to get hit, multiplied by a constant, k, of around 1.05; I arrived at this number by understanding that from each θ_O, there was a max of 60˚ to be added or subtracted, and since the time difference was the only other variable at my disposition I found it logical to multiply these two numbers and divide them by some number that would give me the best results. This number turned out to be 57, so 60/57 gave me the constant k of 1.05.

At last, having the logic to determine which mic got hit fisrt, whether to add or subtract from the θo, the absolute time different between the second and third mics, and the constant k, the angle of arrival could be calculated using the following equation:

$$\theta = \theta_o + \left(|T| * \left(\frac{60}{57}\right) * (+/-)\right)$$

Readings and Hardware

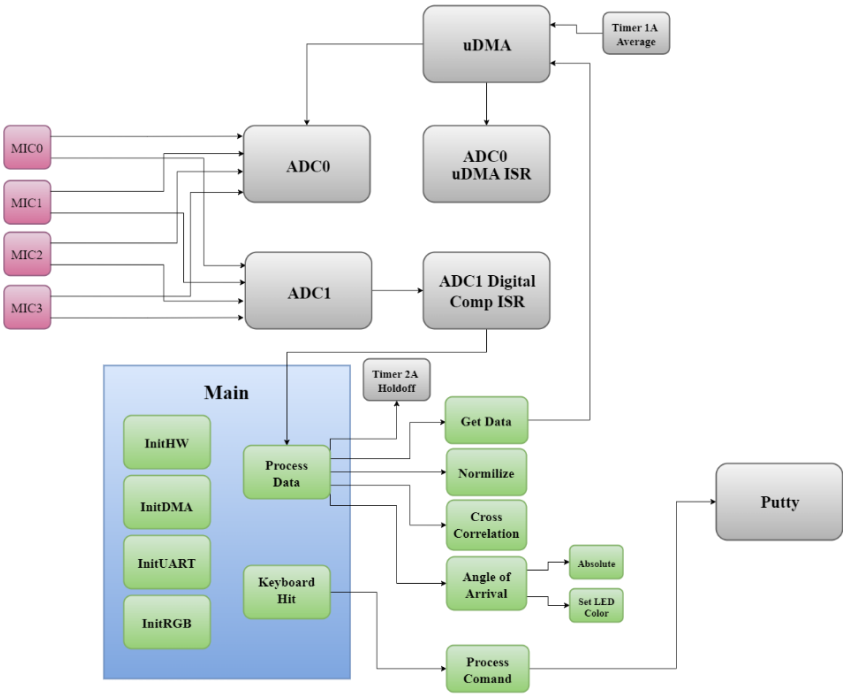




Failings/What Could've Been Improved

Since the end goal of this project was not entirely met with my approach there are several aspects that could've been improved upon, or straight up done differently altogether. Regarding the hardware, everything seemed to be working fine and the microphones picked up sound properly. There was no need to remove noise from them, so the physical components worked accordingly. Now, as for the software side of things, the uDMA could've been configured as a Single Transferred as supposed to Ping-Pong, since the former would've accomplished the same things with a shorter configuration. Testing methods could've been completely different since I mostly relied on snaps, claps, and/or loud sounds to test the functionality of the device, when I should've been testing with a regular voice; the goal of the device was to compute the angle of when it detected conversation. Having mentioned this, with the approach that I took on this project a filter should've been implemented to filter out echoes that would bounce back into the mics, since these echoes would create distorted reading. Moreover, according to TA a mid-band configuration should've been done with the digital comparators, as supposed to high-band. The reason being that in high-band a loud sound is required to trigger an event, while in mid-band a constant sound, like a voice, in between an upper and lower limit would trigger an event.

Software Block Diagram



Hardware Circuit Diagram