Embedded II Project Report:
Sound Detection Machine

Written by Eduardo Ramos
1001834020
5/2/2024

CSE 4342
Professor Jason Losh
Section 004

Introduction:

The processing of signals and what we can gather from them have been an imperative pillar of modern engineering. The ability to transmit information over long distances is not only critical to our daily operations but serves as an example of the intriguing phenomena that is our world. To explore this principle a bit further, the construction of this project was an attempt to explain one simple question. If possible, how could one use a sound wave to produce informative data?

Solution Approach:

To put our solution in perspective, we designed a multi-directional sensitized microphone machine which has 4 equidistant microphones. The purpose of these microphones is to gather relative data of the sound amplitude to which we can use to derive their direction of the source. To convert the electrical signals into data we can process, the Tiva M4F Microcontroller was used as our device choice. Now, the Tiva Board's Analog Digital Converter offers a method to collect data from each of the individual microphones. However, to determine what amplitude is equivalent to a significant event had some ambiguity we had to resolve. For starters, there were two methods of determining a valid event from either the analog or digital side. Using an analog comparator would provide a robust method of testing for a valid event given that it only requires an electrical signal. However, analog signals tend to be noisy in nature, especially given the circuitry involved with the project's construction. On the other hand, a digital comparator offers a noise-free option for receiving the input of the microphones. However, that would increase the size of our program, which could pose portability issues. Taking all of this into consideration, it was decided that a digital comparator would give us better results since electrical interference would not pose an issue. To operate the digital comparator, two threshold levels must be provided as dictated by the datasheet, which were calculated through experimenting with different values and calibrating accordingly. In the end, a lower threshold value of 0x005 and an upper threshold value of 0x120 were selected as the most appropriate for our design. In so, if any of these Digital Comparators receive a signal above the upper threshold, a valid event is found. Before we interface the microphone signals into our microcontroller, it was necessary to amplify them. Thus, an LM2902 amplifier chip would be required to gain our signal to appropriate levels, giving us better results. Once we have constructed the circuitry needed to convert signals into voltage, our microcontroller takes over the calculations.

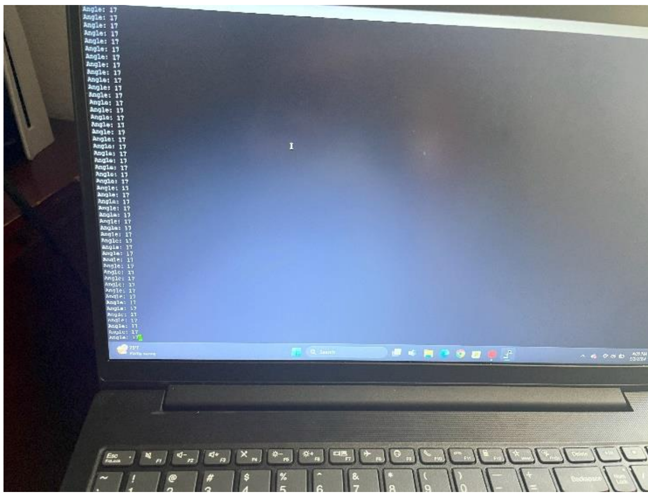
Protocols and Peripherals:

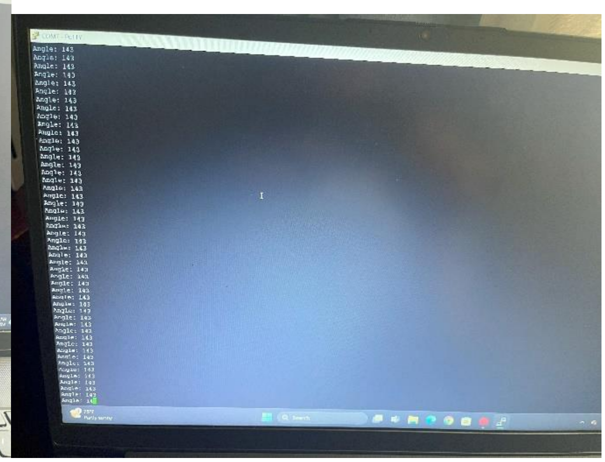
As stated previously, the Analog Digital Comparator will sample our data given a certain configuration. In this case, we connect each of our microphones to Analog Inputs 1,2,4, and 5 in that respective order. From there, we configured the ADC to sample at a rate of one sample per microsecond with a FIFO buffer size of four. This configuration was applied to the ADC Module 1 to serve as our sampling device. As for Module 0, we would have the same configuration with some additional changes. To enable our module to use Digital Comparators, we must connect each one of them to a specific input. Once achieved, we set the threshold values and enable both modules with interrupts placed as well. To determine the angle of arrival for our sound, we need a buffer of sorts. Thus, we would enable the Direct Memory Access module or DMA for shorts. This module, when configured properly, will act as the delivery mechanism to transfer input data to be stored into certain parts of memory. In so, we remove the burden of the microprocessor which will perform our correlation calculations at a faster rate. Once these peripherals are enabled, we can sample our data when a valid event is triggered, which then the DMA will transfer to our designated buffers. However, it is important to note that the interrupts for our microcontroller have a slight delay, which we must consider when sampling our data. Thus, a slight delay of $\tau/2$ will be needed to capture our valid event, where τ is the time it takes for sound to reach the farthest microphone.

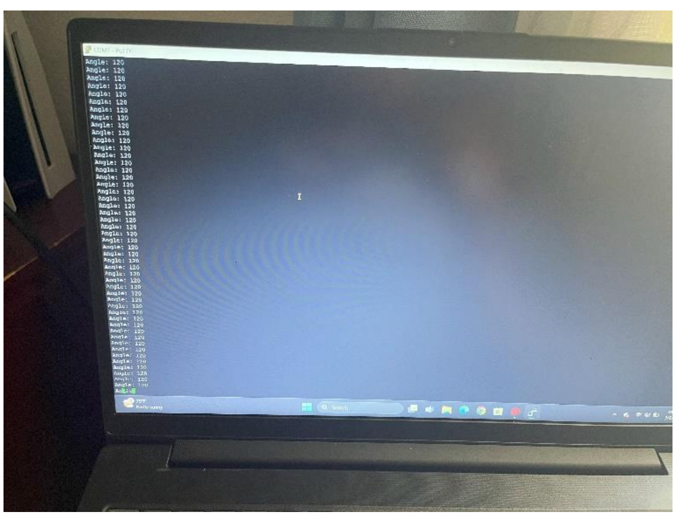
Algorithm and Angle Calculations:

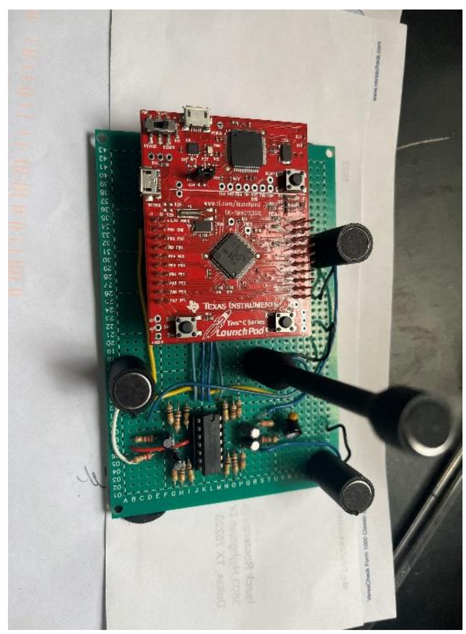
At this point in the design process, we have obtained our digital values from the ADC module stored into our buffers. To make use of this data, we will cross correlate our different microphone samples with a sliding window. This sliding window will have a snippet of sorts of the data obtained from another microphone. The objective is to multiply the value of the first microphone sample at the current index with the value of the sliding window. The value of the sliding window will change as it depends on the index of the first sample and the current index the sliding window is. In so, we obtain the correlation samples, in which we can find the index of the highest value. In this example, we had three correlations sample arrays, each with their own highest correlation index. Out of these three, the biggest value tells us that the microphone that received the sound first is could be either microphone corresponding to the correlation sample. For example, if the biggest index corresponded to the correlation sample of microphone 1 & 2, then either microphone 1 or 2 received the sound first. Thus, knowing this, we would infer that microphone 3 received the sound second. From this information, we would compare the other two index values in which a smaller value points to the actual microphone who received the sound first. Going back to our example, if the index of correlation 1 & 3 is smaller than the index of correlation 2 & 3, then microphone 1 would have received the sound first, then microphone 2 and then microphone 3. Now that we know which microphone received the sound first, we can use its respective angle as a start for our angle calculation. To calculate the complete angle, we would assume that our k_1 constant to be 1, which was assumed given time constraints. After deciding on the constant, we would take the correlation index that helped determine the actual microphone to receive the signal first and add/subtract depending on the direction of the microphone of the second correlation index. Through all these calculations, we have determined our angle of arrival.

Results:



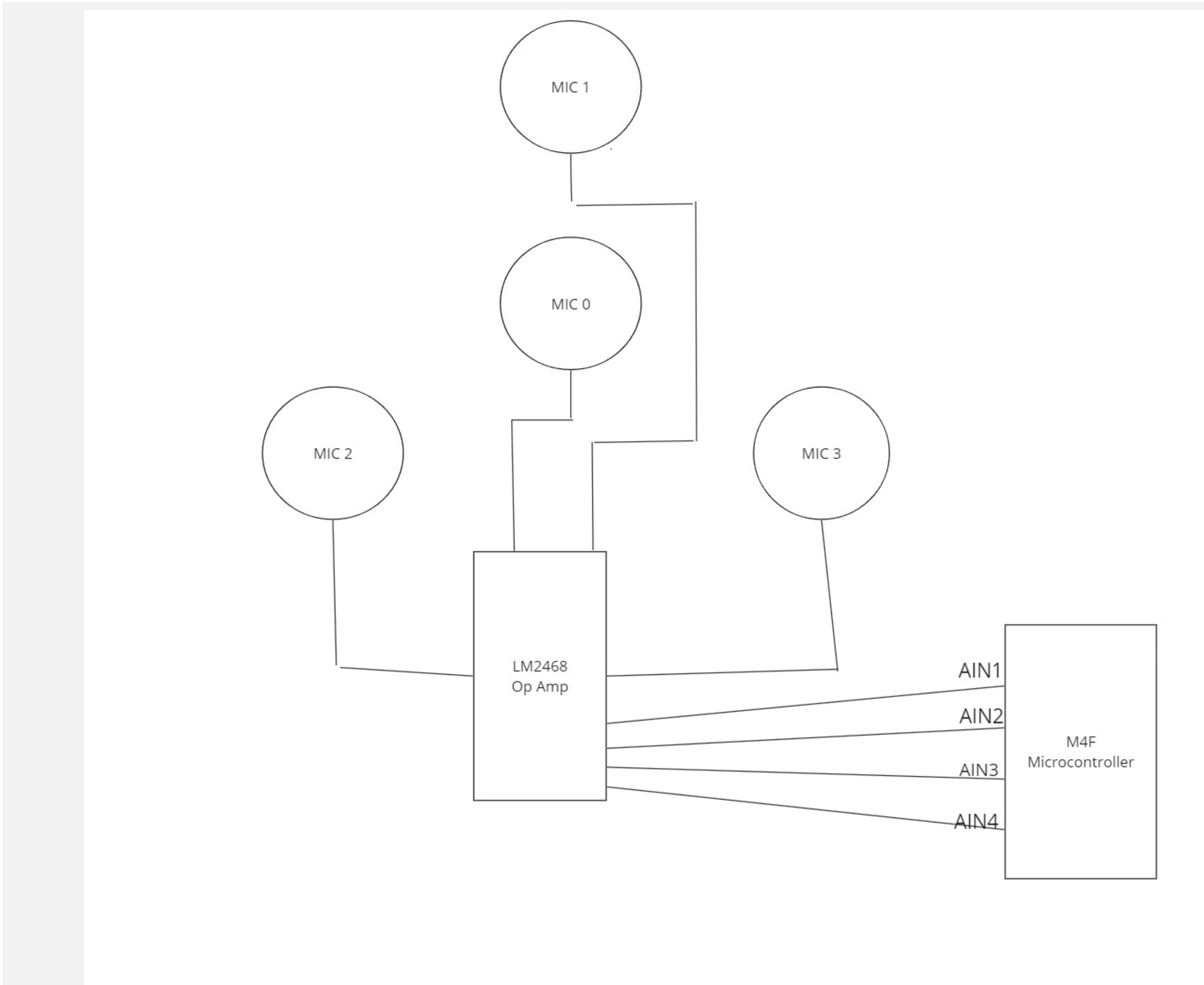


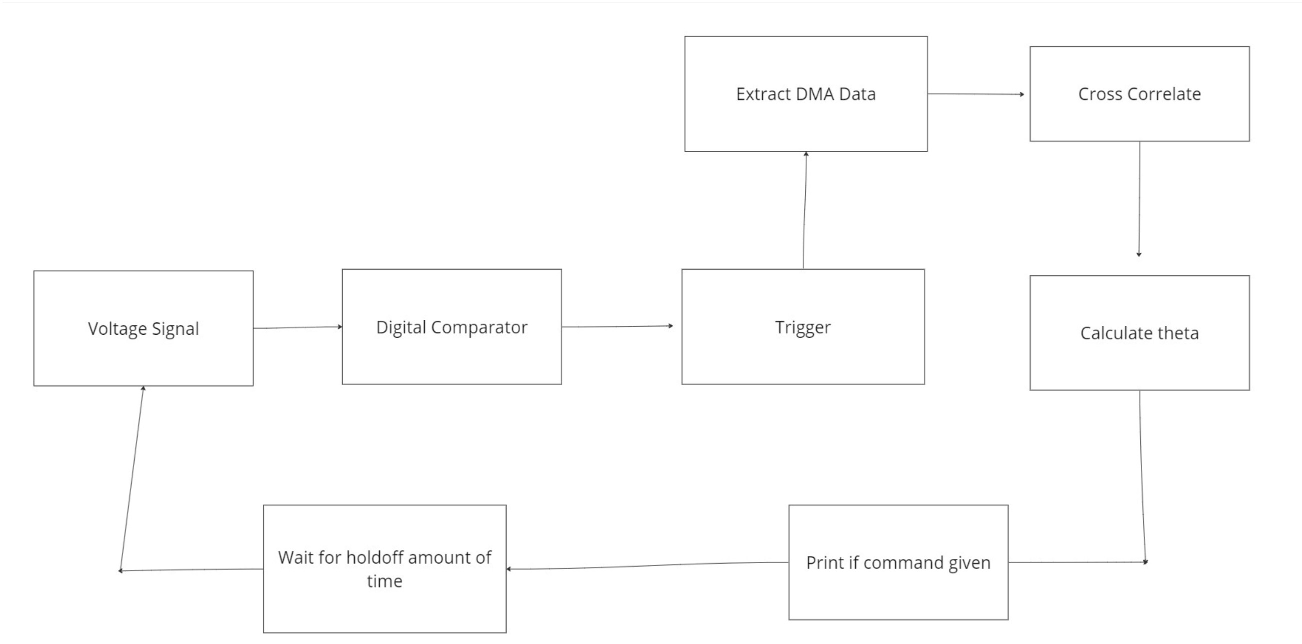




Design Failings:

Upon completing the project, it was found that the current methods of calculating angles did not yield results within our accuracy expectations. While there could be many reasons for this, one of the most prominent is a lack of soldering organization. During construction, the organization of the circuitry fell below expected levels, which could have contributed to unwanted electrical noise. Yet knowing that, it could be concluded that the calculation method chosen for our project design was highly inaccurate, which led to an increase in errors within our predictions. Thus, if this project is to be replicated, a better K_1 constant would be needed as to properly correlate the distance between our base angle and the actual angle of the detected sound. Another improvement that could be made for this design is the implementation of bypass capacitor, which could reduce the noise caused by circuity in proximity. To add more protection against electrical noise, twisted pair wires can be used to cancel the electromagnetic field effect that wires may produce. Despite these flaws within our design, we were able to determine the angle of arrival with some extent of accuracy. However, if this prototype is to serve as base for a marketable product, intensive improvements must be made to ensure the accuracy of the calculations are within acceptable limits.


MIC 1
MIC 0
MIC 2
MIC 3
LM2468
Op Amp
AIN1
AIN2
AIN3
AIN4
M4F
Microcontroller


Extract DMA Data
Cross Correlate
Voltage Signal
Digital Comparator
Trigger
Calculate theta
Wait for holdoff amount of time
Print if command given