Jonathan Wapman

Abstract

Using two microphones, it is possible to locate the direction of a sound source. In this project, two microphones are separated on opposite sides of a meterstick, and a LabView Virtual Instrument measures the time difference it takes for the sound signal to reach each microphone. The VI determines if the sound signal is above the cutoff amplitude, calculates the incident angle of the sound source, determines the relative intensity of the signal in each channel, and then sends this information to a microcontroller. The microcontroller points a servomotor in the direction of the sound source and blinks red and green LEDs to indicate the relative intensity of the sound detected by each microphone. This detection method is generally accurate to within several degrees for a distinct noise such as a clap, but has limited usefulness for longer-duration sounds.

Introduction

The purpose of this project is to capture a signal from a pair of microphones, process the digital data, identify the direction of a sound source, and point a servomotor at the sound source. For the microphone system, I soldered a microphone and an amplification circuit to two circuit boards. In order to differentiate sounds between the two microphones, I separated the microphones so that the sound's travel time from the source to the microphone would be significantly different. To accomplish this, I mounted the microphones on either end of a meterstick, with 1.15m between the two microphones.

Next, after capturing the audio from each microphone, I passed the digital signal through a bandpass filter. In the UI, the user has two input boxes where they can select the high and low cutoff frequencies for the bandpass filter. If the user knows the general range of the frequency they are looking for, the bandpass filter allows them to isolate that range of frequencies. For example, a clap is generally in the range of 3000-5000 Hz. A bird whistle is in the range of 1500-1800 Hz.

After isolating the signal, the VI determines the direction of the sound. To do so, it looks for the peak value of each audio channel and determines the time difference between the peaks in each channel. Using this time difference, the VI calculates the direction of the source relative to the microphone system. To reduce error from noise, the VI only looks for peaks above a certain cutoff amplitude (entered by the user). The VI then determines the intensity of the sound in each channel based on how many times greater the amplitude is than the cutoff amplitude. Finally, the VI creates a string of information including the angle and each intensity and writes that information to the serial port of the microcontroller.

The microcontroller continually checks for new serial data. If new data is available, the microcontroller parses the data into three variables containing the angle to rotate the motor to and the number of times to blink each LED. The microcontroller moves the servomotor to the desired position and then blinks each LED the number of times specified by the serial string using two sequential loops.

Finally, an extra feature the project includes is the ability to record audio from the microphones to a file. When the user clicks the "record" button, the VI saves sound data from the microphones to a .wav file in the computer's filesystem, which can be opened by any third party audio player.

Description

In order to localize the sound, I separated each microphone by mounting them 1.15m apart on a yardstick. This created a measurable difference in the time it took for the audio sound to reach each microphone. The VI captures the signal via the myDAQ's audio input port at 100kHz and 20k samples per second, which it then uses to create a digital signal. The VI sends this signal through several filters and then sends the signal into a Matlab script node within LabView. The script takes the amplitude vs time data and separates it into two arrays containing the left and right channels. It identifies the peak values of each channel, the index of each peak, and the number of samples between each peak. Since the program takes 1 sample every 0.00001 seconds, the program can calculate the time difference in sound arrival time by multiplying the number of samples between each peak by the time per sample. After identifying the peaks and time difference, the script checks to make sure the peak is above the threshold amplitude (entered by the user in a text box), rather than just background noise. If

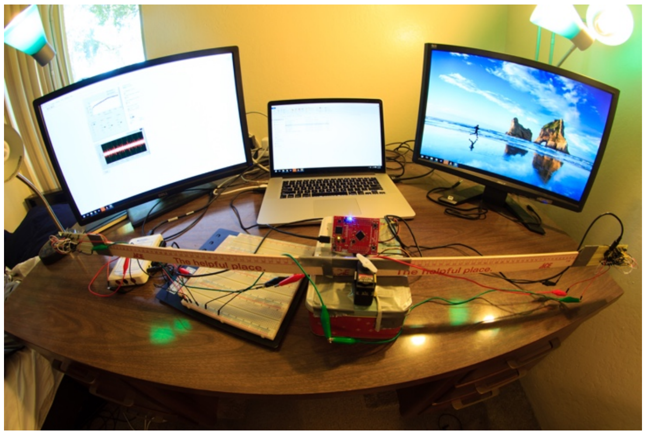


Figure 1: Microphone Setup

the script determines that a clap occurred, it uses the following formula to determine the angle from the center.

$$\theta = sin^{-1}\left(\frac{343m}{s} * \frac{\Delta t}{d}\right)$$

In this formula, theta represents the angle from the center that the sound occurred (from -90 degrees to 90 degrees), d is the distance between the two microphones, and 343m/s is the speed of sound. I then shifted the angle to a value between 0 and 180 degrees so it could be written to the servomotor, using the equation

$$\theta_{new} = -\theta_{old} + 90$$

Finally, the program uses the maximum value of each peak to determine which sound intensity to indicate for each channel using the LED's. An intensity of 1 corresponds to the threshold amplitude, 2 to twice the threshold, and 3 to four times the threshold. These values are stored in the leftIntensity and rightIntensity variables.

After determining which direction the sound came from, the LabView writes the information to the TIVA microcontroller. The Matlab script uses the strcat() function to create a string of the form 'angle, leftIntensity, rightIntensity'. Within the VI, a counter continuously counts from 1 to 15 and repeats. On the count of 15, the LabView program writes the string to the Serial port of the microcontroller. By having LabView write the data on a counter, the VI gives the microcontroller time to move the servomotor and blink its LEDs without becoming overwhelmed with data from LabView.

Finally, the last step is to move the servomotor to point at the source of the sound. In a loop, the microcontroller continually checks to see if serial data is available. If it is, it parses the input into variables storing the angle to rotate to and the number of times to blink the red and green LEDs. To do so, the loop calls the readStringUntil function three times, reading until it reaches a comma.

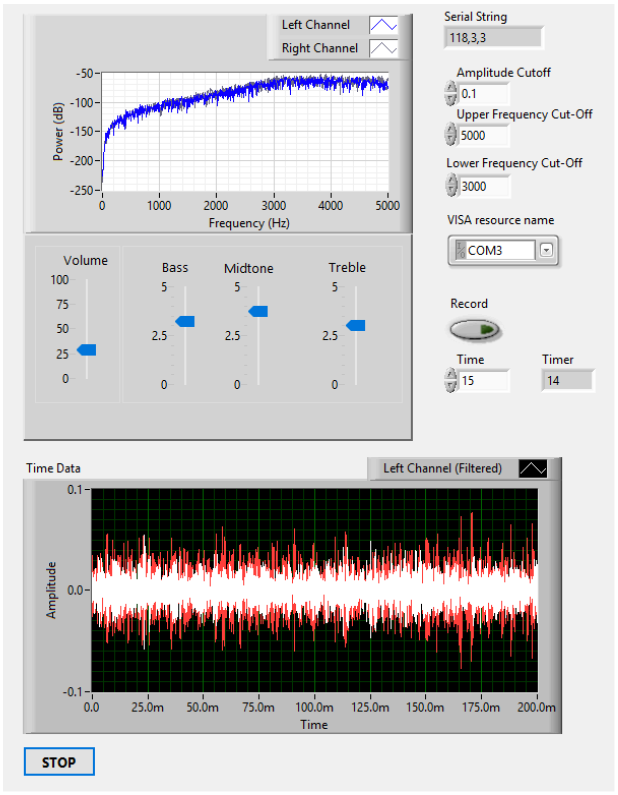
```
input = Serial.readStringUntil(',');
```

After it has the string, the TIVA converts the string using a function formatted as

```
leftIntensity = input.toInt();
```

The TIVA then rotates the servomotor to the new angle by calling the servo.write(angle) function. Finally, the TIVA determines how many times to blink the LEDs. The leftIntensity and rightIntensity values determined by the Matlab script correspond to how many times the TIVA should blink each LED. Using two sequential loops, the TIVA first turns the red LED on and off as many times as specified, and then turns the green LED on and off. After it has finished blinking, the TIVA turns off both the red and green LEDs and turns on the blue LED to indicate that it is ready for new data.

One of the features implemented into the program was stereo audio recording. The first time the VI runs, LabView creates a new blank .wav file to store audio data. When the Record switch is clicked to the “on” position, the signal is written to the file. To do this, the filtered signal from the myDAQ is first sent to an Align and Resample filter. Due to the refresh rate of the myDAQ, the unchanged audio is half the speed it should be. The Align and Resample filter doubles the sample rate of the sound and passes the sped-up signal to the file to be written. The user can open the audio file in the computer’s filesystem and play it using any third-party audio software. In order to reduce noise and emphasize different ranges, the user has several filtering options. First, the user can select a range of frequencies to isolate by entering the values in two text boxes labeled “Upper Frequency Cutoff” and “Lower Frequency Cutoff”. These values are applied to a bandpass filter, which filters the signal from the myDAQ before it is passed on to the rest of the VI. Next, the user has 4 slider bars corresponding to the overall volume, and the volumes of the bass, midtone, and treble ranges. By using these volume sliders, the user can reduce unwanted background noise or isolate the desired frequency ranges. In the VI this is accomplished by splitting the signal into three ranges using



Serial String 118,3,3
Amplitude Cutoff 0.1
Upper Frequency Cut-Off 5000
Lower Frequency Cut-Off 3000
VISA resource name COM3
Time 15
Timer 14

Figure 2: VI User Interface

three bandpass filters, and then amplifying the signals by multiplying each one by the corresponding slider's value.

In general, this system works well for some vocalizations, but poorly for others. It is strongest in identifying abrupt sounds such as claps. This is because the sound has an easily identifiably peak which the program can use to calculate the time difference between each event. However, this method has difficulty identifying the source of a drawn-out sound that does not have distinct maxima. Early attempts involved attempting to find the source of the audio by comparing the average amplitudes of each microphone. This was not as effective as desired, since this method has limited accuracy beyond simply determining whether the sound is to the left or the right of the microphones. This problem could be overcome with more accurate amplifier circuits and microphones, but was impractical for this particular experiment. Another issue that the program occasionally has is if the vocalization is towards the very beginning or end of the myDAQ's sample. In this case, one or both channels may be cut off, so the program may determine an incorrect source direction because it only receives a peak signal from one microphone. This can be somewhat overcome by checking that both the left and the right channels have peaks above the threshold amplitude. Finally, the microphone works best when the sound source is close to the microphones. The closer the source is to the microphones, the greater the time difference between the sound's arrival at each microphone. This problem could be solved for greater distances by placing the microphones farther apart or by sampling the source more frequently.

Resources and Acknowledgements

The equation to determine the direction of a sound source comes from https://www.andrew.cmu.edu/user/jbresson/Angle Detection Based on Interaural Time Difference.htm. My project used portions of "Continuous Serial Write and Read.vi", "myDAQ Audio Equalizer with Noise Test.vi", and "myDAQ_Audio_Template.vi". I also utilized LabView and Matlab online documentation. A video of my project can be found at https://www.youtube.com/watch?v=RJ8DV89I57g and my source files can be found at https://ucdavis.box.com/s/k12qpqk0dx8drtmh36cn7ouppes8p86p