Aerial Drone Tracking with Passive Sonar

Advisor
Dr. Martin Siderius

Sponsor
Metron Scientific Solutions
John Gebbie
Jeffery Hoffman

Portland State University

Winter-Spring 2018

Surya Ravikumar, Mitch Pyle, Yusheng Tian, Dylan Zodrow, Brian Milanek

Table of Contents

Abstract

The purpose of this project is to develop a system to track aerial drones by listening for the sound waves they emit. The system uses audio input from multiple microphones to process data and display the result. Distance between multiple microphones in a microphone array is exploited to determine the direction where the sound originated from – known as beamforming. A single direction, along with elevation, is output on a PCB using LEDs out of 8 possible directions with a 45-degree range between them. This report gives a detailed description of the requirements, assumptions, procedure, limitations and bill of materials. This project was proposed and sponsored by Metron Scientific Solutions and advised by Dr. Martin Siderius.

Acknowledgements

This project would not have been possible without the support of many people. We would like to thank Metron Scientific Solutions for proposing and sponsoring the project, as well as John Gebbie and Jeffery Hoffman of Metron, and Dr. Martin Siderius for their guidance, support, advise, input and arrangements needed for the success of this project. We would also like to thank Elizabeth Kusel for providing us tools and help in the initial stages of the project to collect drone audio data.

Introduction

The purpose of this project is to develop a system to track aerial drones by listening for the sound waves they emit. Since the system uses sound waves, it is important to consider the environment where the system will be used and the type of drone it can detect.

Use of sound waves instead of any other signals simplified the issue. It meant that if the system were to be used in an environment where multiple signals are transmitted and received by numerous devices, our device would not interfere with those signals. Using a passive system further simplified the issue as the system would not have to transmit signals in order to track drones.

Some issues related to using sound waves as primary input to track drones were:

1. Noisy environment
2. Reflective environment
3. In-audible drone sound

The objectives of the project were:

1. Learn Digital Signal Processing to process audio
2. Design system and microphone array
3. Develop algorithm to process input
4. Decide output format to display direction accurately and comprehensible manner

This document provides the process, design decisions, limitations, challenges and further improvements.

Specific Requirements

The section lists specific requirements for the Aerial Drone Tracking System.

User Requirements:

1. Identify when a drone is near, locate the drone's direction, and indicate to the user the direction the drone is located.
2. Portable and durable enough to be used as a demonstration model
3. Accurate enough that the indicated direction is close to the actual direction of the drone
4. Operate independently after initiation by the user

System Requirements:

1. Identify when a drone is operating within 100m under suitable test conditions
2. Accurately located a drone's direction to within $\pm 5^{o}$ within range of operation
3. Handle a Signal to Noise Ratio (SNR) of 60dB
4. Location must be refreshed in <1 sec
5. Any error states must be resolved by a restart
6. Durable enough to be assembled/disassembled a minimum of 10 times
7. Entire system must weigh less than 20 kg

Interface Requirements:

1. Directional output must be visually intelligible to a person within 5m
2. Operable by a person with basic knowledge of the system
3. Only necessary input is stop/go

System Design

The system uses a circular microphone array, with an 8-inch diameter, consisting of 8 microphones, each at 45° separation. This spacing gives the microphone enough distance to experience lag between them, as well as limiting the max lag experienced by two microphones be equal to:

$$\frac{8\ inches\ \times \frac{0.0254\ meters}{1\ inch}}{343\frac{m}{s}}$$

$$= \quad 0.59\ milliseconds$$

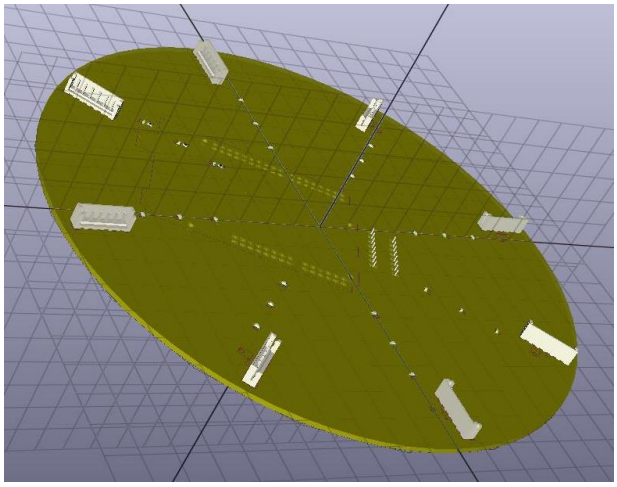


Figure 1. Microphone array on the PCB - CAD image

8 channels were used since that gave the algorithm more than enough data to localize. This was also the max number of channels supported by the FRDM K64F microcontroller. Since the ICS 52000 microphones are daisy chained, sending a signal to the first mic in the chain is enough to receive data from all other mics in the array (each mic sends out a WSO signal to the next mic in line when it recieves a WS signal). The microphone ports on the PCB were designed such that microphone boards can be easily replaced – not permanently attached to the board, and easy to remove and connect

It was decided that LEDs would be a feasible option to display the final result. 8 directions, same directions as 8 mics, were chosen to potray the approximate direction. Along with direction, 3 leds in each direction were included to represent angle of elevation of the drone with respect to the system.

The PCB was designed such that the pins that connect various elements on it can be directly connected (inserted) in the ports on the microcontroller. This made the system easier to handle gave the user the comfort of not having to deal with hanging wires.

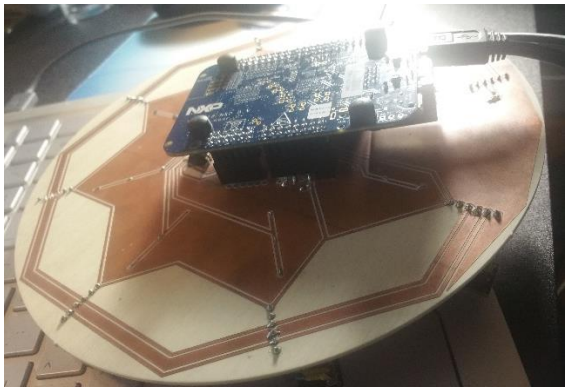


Figure 2. Microcontroller attached to PCB

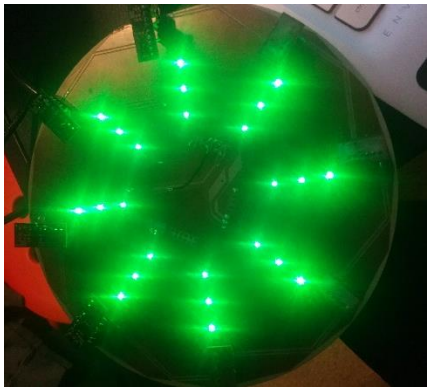


Figure 3. LEDs that show direction and elevation (NOT showing a particular direction and elevation)

The whole system consists:

1. *A PCB*
    a. 8 ICS 52000 mics
        i. 8 ports on board that microphones are inserted in
        ii. Daisy chained
        iii. Common data and clock line
        iv. Frame Select bus from each mic to next mic
    b. 24 LEDs
        i. Controlled by a 3 to 8 mux to select specific direction
        ii. 8 to show direction
        iii. 3 in each direction to show elevation
2. *A Microcontroller*
    a. FRDM K64F
    b. Needs a 5V power source
    c. Connects directly to PCB

Data Processing

In general, cross correlation measures the similarity of two signals (series). It results a value that tells how much a signal is shifted along the x-axis.

$$c(t) = \int_{-\infty}^{\infty} f'(\tau)\, g(\tau + t)\; d\tau \quad [1]$$

If ***f*** and ***g*** are two signals in frequency domain, where ***f'*** is a complex conjugate of ***f***, and if one of the signal is just a shifted signal of the other, cross correlation can be used to find how much the shifted signal is shifted by. The formula above takes the conjugate of signal ***g*** and slides it along the x-axis calculating the integral of the product of ***g*** and ***f'***. When the positive areas are lined up, it adds to the integral, and opposite when negative areas line up. This results in a single peak that is the difference between the signals.

Audio data from mics are cross correlated to measure the difference between the signals.

1. Audio data from 8 channels are sampled at 48000 Hz.
2. One channel is used as reference and is correlated with other 7
    a. Take FFT of both channels
    b. Multiply the result of conjugate of the FFT of the reference channel with FFT of the channel that is being correlated with
    c. Divide the result with the absolute value of the result to suppress certain values
    d. Inverse FFT the result and find a peak
    e. Use the index of the peak to find delay between the two signals in samples
    f. Divide the resulting sample by sample rate to get lag between the reference channel and the respective channel

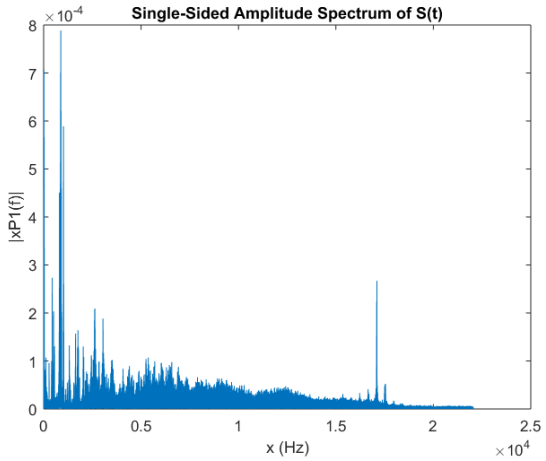


Figure 4. Result of FFT of audio signal (imaginary part removed) – result of test case 1 (in testing and results)

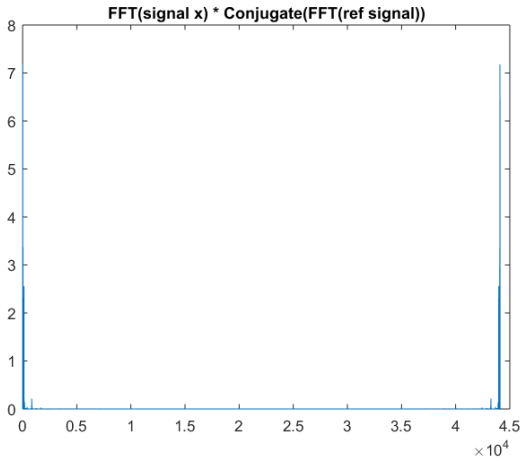


Figure 5. Result of multiplication of two signals in frequency domain – result of test case 1 (in testing and results)

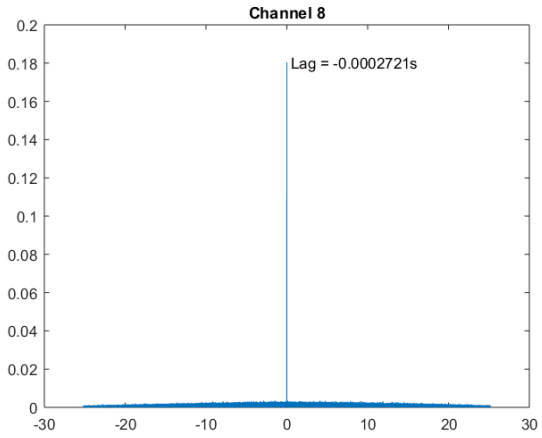


Figure 6. Final result showing single peak and lag – result of test case 1 (in testing and results)

```
Channel 1: 0.0000000000
Channel 2: -0.0000907029
Channel 3: -0.0000907029
Channel 4: 2.6861224490
Channel 5: -0.0005668934
Channel 6: -0.0006349206
Channel 7: -0.0005215420
Channel 8: -0.0002721088
Led to turn on: 6
```

Figure 7. Result of real data with respect to channel 1 – result of test case 1 (in testing and results). NOTE: channel 4 was broken when recording

Minimum lag (most negative or 0) is found from the obtained result and LED for that direction is turned on. Elevation is calculated using:

$$abs(lag\ between\ ref.mic\ and\ mic\ directly\ opposite\ to\ least\ lagged\ mic - \ least\ lag)$$

Where result of above equation - $\boldsymbol{x}$:

$\boldsymbol{x}$ > 0.592 ms → invalid
0.592 ms > $\boldsymbol{x}$ > 0.395 ms → same level as board
0.394 ms > $\boldsymbol{x}$ > 0.197 ms → mid-range
0.197 ms > $\boldsymbol{x}$ → directly above board

Testing & Results

A. Accuracy

Multiple cases were used to test the algorithm, and the algorithm was able to accurately localize the drone. Some of the cases are included below

1. *Stationary drone (original data)*
   a. Drone was approximately set in front of microphone 6 and the result below shows that the algorithm correctly calculated drone was closest to the direction pointed by mic 6.

```
Channel 1: 0.0000000000
Channel 2: -0.0000907029
Channel 3: -0.0000907029
Channel 4: 2.6861224490
Channel 5: -0.0005668934
Channel 6: -0.0006349206
Channel 7: -0.0005215420
Channel 8: -0.0002721088
Led to turn on: 6
```

Figure 8. Result of real data with respect to channel 1 (NOTE: channel 4 was broken when recording)

2. *Moving Drone (original data)*
   a. Drones initial position was close to mic 6 and moved 90° to the right (towards mic 8) and back. The result below shows that the algorithm was able to follow the drone.

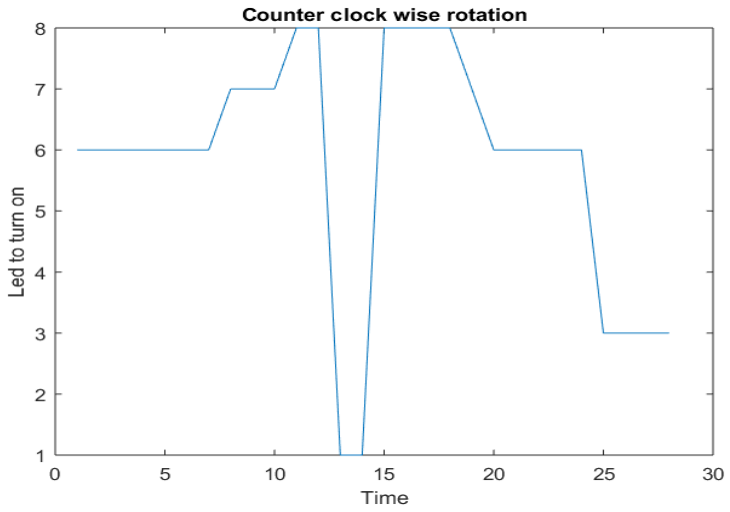


Figure 9. Graph showing mic that represented the closest direction of the drone

3. *Moving drone (synthetic 8 channel data using original 1 channel drone audio)*
    a. These cases where created synthetically modelling the drone movement giving drone enough time in front of each mic. Expected results and calculated results are included below (all lags in seconds)
    b. In expected results table, few values are recorded as 0s since algorithm rounds extremely small lags to 0, and couple of lags don't match with expected values.

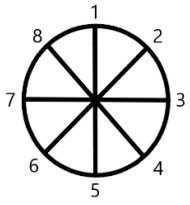
Microphone Number								
1	2	3	4	5	6	7	8	Closest Mic.
0	0.00014523	0.00036175	0.00052371	5.27E-04	0.00036848	0.00015164	2.57E-06	1
0	2.08E-04	4.08E-04	0.00047423	0.00036245	0.00014698	-4.13E-05	-0.00010028	8
0	2.56E-05	-6.70E-05	-0.00022034	-0.00034362	-0.00036772	-0.00027949	-1.28E-04	6
0	-1.54E-05	0.00011347	0.00031834	4.81E-04	0.00049815	3.58E-04	1.52E-04	2
0	-9.35E-05	-1.23E-04	-7.17E-05	3.11E-05	0.00012581	0.0001561	1.04E-04	3
0	-5.67E-06	-7.52E-05	-0.00016725	-0.00022793	-0.00022237	-1.54E-04	-6.16E-05	5
0	-9.53E-05	-1.85E-04	-2.18E-04	-1.75E-04	-8.05E-05	1.06E-05	4.42E-05	4
0	2.20E-04	3.45E-04	2.92E-04	9.68E-05	-0.00011713	-0.00022951	-1.83E-04	7

Table 1. Table showing expected lags with respect to mic 1 at each position

Microphone Number								
1	2	3	4	5	6	7	8	Closest Mic.
0	0.00015873	0.00036281	0.000521542	5.21542E-04	0.0003628	0.00015873	0	1/8
0	2.18e-04	4.17e-04	0.00045408	0.00035351	0.000136054	-4.226e-05	-0.00011140	8
0	2.26e-05	-6.8027e-05	-0.00022676	-0.00034014	-0.00036281	-0.0002721	-1.13378E-04	6
0	-2.267e-05	0.00011337	0.00031746	4.7619E-04	0.000498866	3.6281E-04	1.5873E-04	2
0	-9.070e-05	-1.1337E-04	-6.8027e-05	4.53515e-05	0.000136054	0.00015873	1.1338E-04	3
0	0	-9.0703e-05	-0.00015873	-0.00022676	-0.00022675	-1.587E-04	-6.802721e-05	5
0	-9.07e-05	-1.814E-04	-2.2676E-04	-1.5873E-04	-6.8027e-05	0	4.535147e-05	4
0	2.2675E-04	3.4013-04	2.9478E-04	9.07029e-05	-0.00011338	-0.0002267	-1.814E-04	7

Table 2. Table showing calculated lags with respect to mic 1 at each position

To calculate elevation, min lag is subtracted from lag of mic directly opposite to it:



If mic 1 is calculated to be mic with least lag, elevation:

abs(lag of mic 5 with respect to reference mic – lag of mic 1 with respect to reference mic)

Figure 10. Mic array with mic numbers

B. Real time performance

1. *Data Processing*

    a. FRDM K64F has a 120 MHz CPU frequency. This was utilized effectively to process 512 samples x 8 channels worth of data 10 times and produce an output every second. Board took some set up time when powered on causing the first output to take ~2 seconds. However, consecutive outputs took just below a second (tested with synthetic data).

    b. If 2048 samples x 8 channels worth of data were to be used, and an output value was produced every single iteration instead of every 10 iterations, the system will work with a refresh rate of < 100 ms (tested with synthetic data).

2. *I2S module*

    The I2S module on the board took ~10 milliseconds (sampled at 48000 Hz) time to read in 512 samples from 8 channels (read 0s on 8th channel – mentioned in issues faced).

3. *LED module*

    When the output was calculated, LED pointing the direction and angle of elevation immediately lit up – no delay (tested with synthetic data).

Issues Faced

Even though the results above look encouraging, multiple issues forced how the data was to be processed.

1. *Noisy environments*

   Since the system uses sound as its input, sound emitted by the drone could be muffled in a noisy environment. Also, the system does not specifically look for the sound of a drone, but instead tries to localize any sound it hears. This property of the system helps the user to track any sound, and the same property is a drawback in a noisy environment as it can't extract/listen for specific sounds. In-audible drone sound in clean environment also causes the same issue as the system cannot read in valid audio data.

2. *Reflective environment*

   In a reflective environment, any sound wave is reflected by any object in the area. A reflected wave is similar to the original sound wave but can approach the microphone array from a different angle (depends on angle of reflection) causing the system to think the sound wave originated from the drone. This leads to incorrect localization. Moreover, the lag between the original sound wave and the reflected sound wave is most likely to be greater than 0.59 milliseconds (calculated above). To combat this issue, the algorithm looks for a peak only between ±0.59 milliseconds lag between the two signals.

3. *Accuracy with respect to number of samples processed*

   Number of samples used to cross correlate channels was a significant factor in the accuracy of results. More the number of samples, more accurate the results. Ideally, with a refresh rate of one second, just below a second of samples will lead to extremely accurate results. However, the DSP functions did not allow to use more than 2048 samples (about 42 ms worth of data). In theory, 42 ms was more than enough data to cross correlate. However, the results did not always match the expected result. Number of methods were tested to see how it affected the result:

   a. *Add a bandpass filter*:

      Adding a bandpass filter somewhat cleaned up the signal and slightly increased accuracy. However, that was still far from perfect. Moreover, adding a band pass filter required more computation time that affected real time performance.

b. *Add a selective filter*

Certain frequencies were set to zero in the frequency domain right after taking FFT of the signal. This did not require a lot of computation time, but the results were still nowhere close to perfect.

c. *Using mode of multiple iterations*

Instead of working with 2048 samples, number samples per iteration was reduced to 512. The algorithm to calculate the direction and elevation was done 10 times, and the mode of the results during these 10 iterations were taken to be the direction and elevation. Reducing samples to 512 helped with computation speed and keep with refresh rate of 1 second, and the accuracy improved massively.

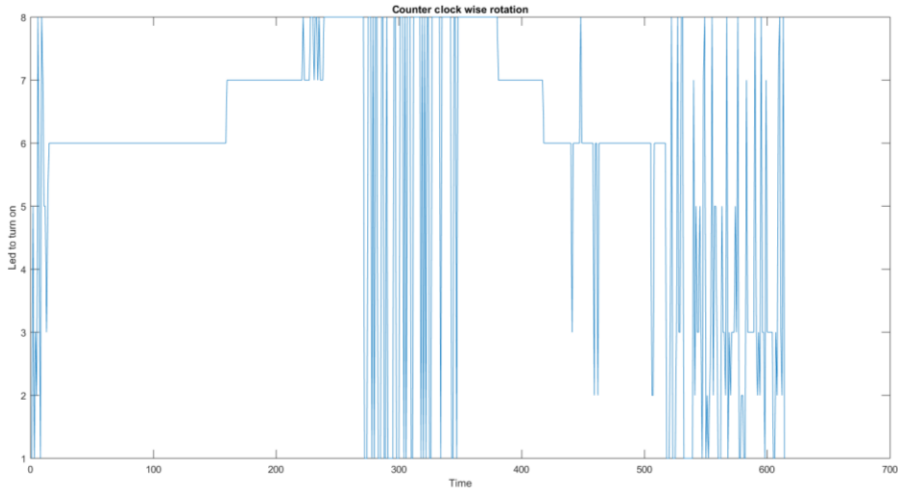


Figure 11. Correlation using 2048 samples (same data as in figure 9 – expected result as in figure 9)

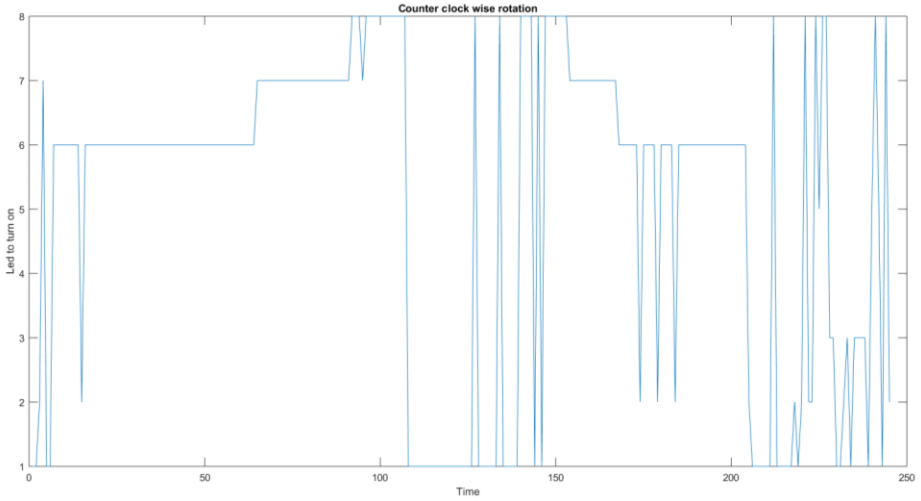


Figure 12. Correlation using 512 samples with averaging (same data as in figure 9 – expected result as in figure 9)

4. *Soldering microphones*

The ICS52000 microphones were extremely tiny, making it hard to solder them onto the boards. Use of reflow over somewhat helped, but the connections were not visible to the naked eye. Some microphones had to be re-soldered since few of them did not generate WSO signals causing the daisy chain to fail, and few did not send out data. Re-soldering the microphones to the microphone board fixed these issues.

5. *Input from 8 channels*

When the mics were daisy chained, the microcontroller was not able to read the last mic in the daisy chain (always read 0s). This was a problem no matter how many mics were connected. Therefore, only 7 mics give valid data for now. This is an unresolved issue.

6. *Clock Rate*

Initial testing used a slower clock rate unintentionally/unknowingly (around 20 MHz) causing the algorithm to produce an output every second when using 2048 samples (single iteration). This was another reason the lead to reducing the samples used to 512. However, the clock was maximized to 120 MHz causing the system to produce an output approximately every 100ms when 2048 samples were used (single iteration), and every second when 512 samples were used (10 iterations).

Improvements

1. *Different computer for processing*

   Even though the microcontroller was an extremely powerful tool, it could not handle more 10000 samples. Further, the DSP functions specific to the board limited the samples to 2048. This resulted in accuracy issue even though the algorithm worked as it was supposed. Initially, own functions were developed adding to the already present MATLAB functions. Therefore, if the microcontroller was only to receive data and transfer it to a more powerful computer like a laptop, more number of samples could be used for accurate results with more computation power for real-time results. However, portability and simplicity of system, and availability of the program on the computer becomes an issue. Or a more powerful DSP that can manage more number of sample could be used to tackle to number of samples issue.

2. *More detailed output - Beamforming*

   With the results from the algorithm, a more accurate representation of the direction can be obtained with some extra calculations – lags can be used to calculate angle of arrival using:

$$\theta = cos^{-1}\left(\frac{\tau_0 c}{d}\right) \quad [1]$$

Where:

$\theta = angle\ of\ arrival$

$\tau_0 = lag$

$c = 343\frac{m}{s}$ $\quad speed\ of\ sound\ in\ air$

$d = distance\ between\ two\ mics$

Even though this method gives an angle – which is much more accurate than flashing closest LED, the accuracy of the result still depends on the lags calculated. If the lag calculated is not precise, final result will not be correct no matter what method is used to show the output.

Results of applying this formula to the stationary drone test case

Calculation between mic 6 and 5:

$\tau_0 = 0.000068072\ s$

$d = 0.0778\ m$

$= cos^{-1}\left(\frac{\tau_0\, c}{d}\right)$

$\theta = 72.535°$

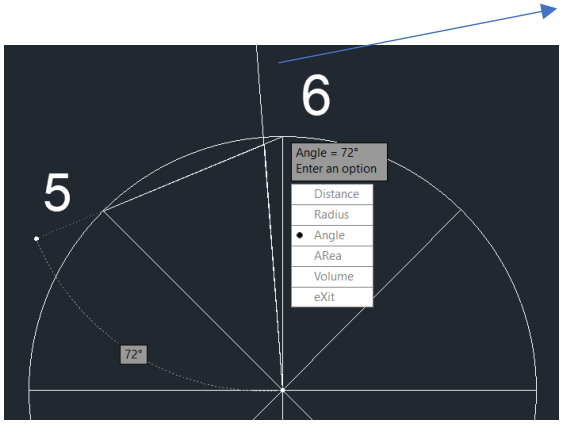


Figure 13. Line showing direction using results from stationary test case

This method needs extra calculations and decision-making steps in the program to decide which lags to use. Calculation do not need a lot of computing power and will not affect real time performance. This angular information could be displayed on an LCD module.

Conclusion

Even though there were few outliers, the algorithm was able to calculate lags accurately when there were enough samples. As the results showed, when the samples processed were reduced, the accuracy of the lags decreased. Using a second's worth of samples (large number of samples) was deduced to be the best option if a processor could handle those large quantities of data. However, this option would still not help in detecting a specific noise to locate in a noisy environment. Therefore, there are multiple details to consider such as real time performance, computation power and complexity before adding to the system.

Board design using 8 channels to get data was more than enough to locate the drone. However, all the data could be used more efficiently to produce a more detailed output.

Inconclusion, the project can be considered partially successful since not all channels on the board produced valid data. With more time and effort, this project would have been successful, and could be improved in many ways.

Bill of Materials

Item	Quantity
ICS 52000 Microphones	8
100 nF capacitors	8
10 uF capacitors	1
LEDs (LTQ39G OSRAM)	24
3 to 8 Decoder (CD74AC138)	1
1k ohm resistors	2
.1” Pitch 2x8 header	2
.1” pitch 2x10 headers	1
.1” pitch 2x6 headers	1
100k ohm resistors	1
1 DSP microcontroller with I2S (FRDM K64F)	1

References

[1] Siderius, M. (n.d.). *Slides for EE 527 Sensor Array Processing*. Portland, Oregon: Portland State University.

Resources

FRDM K64F Datasheet –

https://os.mbed.com/media/uploads/GregC/k64f_rm_rev2.pdf

ICS 52000 Datasheet –

https://store.invensense.com/datasheets/invensense/DS-000121-ICS-52000-v1.3.pdf

SAI/I2S functions –

http://mcuxpresso.nxp.com/apidoc/group__sai.html

CMSIS DSP Library –

http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html

SDK builder for IDE –

https://mcuxpresso.nxp.com/en/welcome

Aerial Drone Tracking with Passive Sonar –

https://github.com/dylanzodrow/Aerial_Drone_Tracking