A practical guide to low-cost, flexible phased arrays implementations (acoustic cameras)

Salim Benchekroun

December 2020

1 Introduction

Throughout the last decade or so, the price of high frequency electronics components (hundreds of Mhz to Ghz) has dropped enough to allow hobbyists to build their own phased array antennas. The following article proposes simple project that would introduce inexperienced hobbyists to the design process. As such, we will focus on microphone arrays as they are far easier to implement.

2 Mathematical Foundation

A simple model for two a two element array can be derived with tools from trigonometry and linear algebra. Expansion to n elements in 3D will be discussed thereafter.

In the following derivation, we will make a few key assumptions about wave propagation, notably we will consider the radiation pattern of the antenna (or microphone) to be isotropic for small values of θ. We will also assume the wavefront is planar at the array. This assumption is reasonable as long the source is far enough (at a distance of 10λ of more).

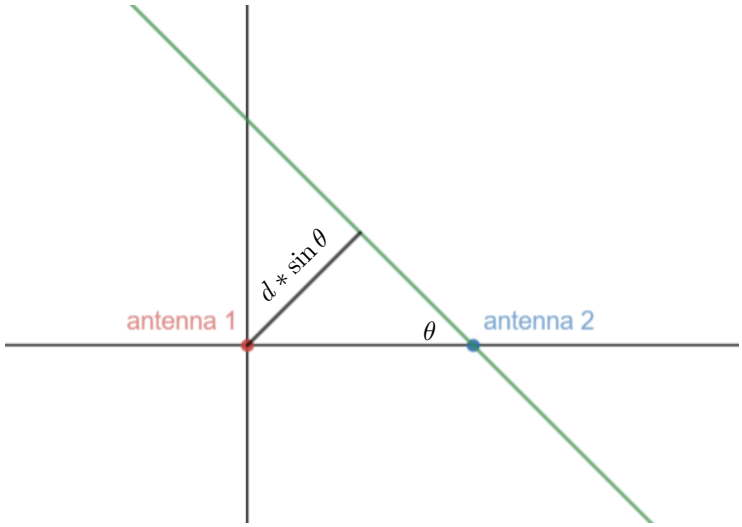
2.1 A linear array with two elements

We will denote variables and parameters of interest as follows:
$\vec{v} =< v_1, v_2 >$: a vector perpendicular to the wavefront
d : the distance between two antennas (m)

θ : the angle between the wavefront (green) and the horizontal axis (rad)

For the sake of simplicity, we will use the terms "antenna" and "microphone" interchangeably.

Here's a simplified diagram.



The wavefront, the line orthogonal to it, and the origin form a right triangle. This means that the time delay between the wavefront hitting antenna 2 followed by antenna 1 is

$$\tau = \frac{distance}{speed} = \frac{d \sin\theta}{v}$$

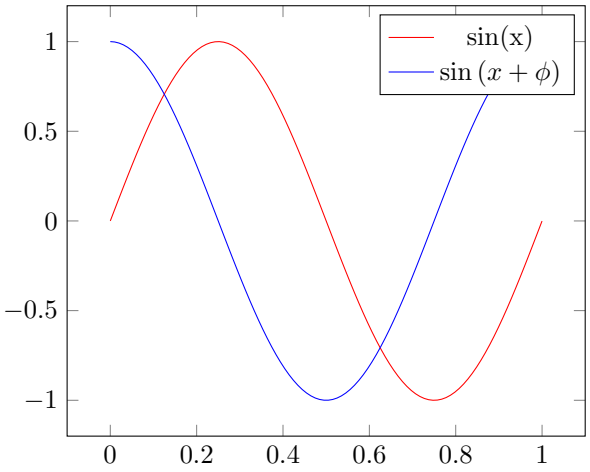
where,
τ : time delay (s)
v : speed of sound at 25 degrees Celsius (m/s)

2.2 Cross-correlation

In the following paragraphs we will give an intuitive introduction to cross-correlation.

The goal of the cross correlation algorithm is to recover the phase shift

between two signals. Once we recover that phase shift, we can find the time delay, and thus we can recover the angle of the incident wave.



3 Practical Example

Before we delve into a specific example, we must broadly discuss the choice of frequency. From the very start, frequency has determined the shape of the project. Higher frequencies usually imply tighter tolerances, but most importantly a timing constraint. The higher the frequency, the smaller the distance between the antennas, and thus the smaller the time delay. Even with down-conversion, capturing enough samples per cycle to feed into our cross-correlation algorithm is a major challenge. For this reason we favored acoustic waves, which are significantly slower than electromagnetic waves.

Acoustic waves cover a broad range of frequencies. To narrow our search of the optimal frequency for a simple project, we can consider a few parameters.

We will set a resolution of θ_{res} degrees as our objective.

The first limiting factor to the precision of θ is the software. Since we are applying cross-correlation to a discrete signal, the number of samples will limit the precision we get on θ.[1]

Let's define a few variables:
f : the frequency of the signal
f_s : the sampling frequency

[1] There are ways to improve the algorithm, but that is out of the scope of this article

To ease development we won't be working from scratch. With an ever growing ecosystem and sufficient resources online the stm32 nucleo family is a great choice. What sets it apart from the other popular boards is its impressive over 1MS/s sample rate.

Even though the adc can reach up to 2MS/s on certain boards of the stm32 family, we will assume the sample rate to be 1MS/s. Also, we will leave room for improvement by making sure that our system could work if we decided to add two more elements on the array. Thus we will assume a $\frac{1MS}{4s} = 0.25\frac{MS}{s}$ sample rate.

So far we know:
v=340 m/s.
$f_s = 0.25$ MS/s

$$\tau = \frac{d \sin\theta}{v}$$

From properties of sines, we know $\frac{\tau}{T} = \frac{\phi}{2\pi}$, where T is the period. We also know $f = \frac{1}{T}$.
Therefore,

$$f\tau = \frac{\phi}{2\pi}$$
$$\tau = \frac{\phi}{2\pi f}$$

Equating the two τ equations,

$$\frac{d \sin\theta}{v} = \frac{\phi}{2\pi f}$$

$$\sin\theta = \frac{v\phi}{2\pi f d}$$

Since $\lambda = \frac{v}{f}$,

$$\sin\theta = \frac{\lambda\phi}{2\pi d}$$

$$\theta = \arcsin \frac{\lambda \phi}{2\pi d}$$

Assuming the distance between each element is $d = \frac{\lambda}{2}$,

$$\theta = \arcsin \frac{\phi}{\pi}$$

This naturally raises the question of angle resolution. By rearranging the previous result, we find $\phi = \sin \theta * \pi$.

Since ϕ must be larger than a certain threshold for our cross-correlation algorithm to detect a phase shift, we can compute angle resolution based on the sample rate.

For a phase shift to be detected, two discrete signals must be shifted by a least 1 sample from each other. To find the range of rads covered by a single sample, we proceed as follows:

$$\frac{rad}{sample} = \frac{rad}{cycle} * \frac{cycles}{samples}$$

By definition, there are 2π rads per cycle, and $\frac{f_s}{f}$ = samples per cycle. Thus,

$$\frac{rad}{sample} = \frac{2\pi f}{f_s}$$

Finally, for ϕ to be detected, it must respect the following condition

$$\phi > \frac{1}{2} * \frac{2\pi f}{f_s}$$

$$\phi > \frac{\pi f}{f_s}$$

$$\theta > \arcsin \frac{f}{f_s}$$

Example: Compute the maximum angular resolution θ if $f = 10kHz$ and $f_s = 250kHz$.

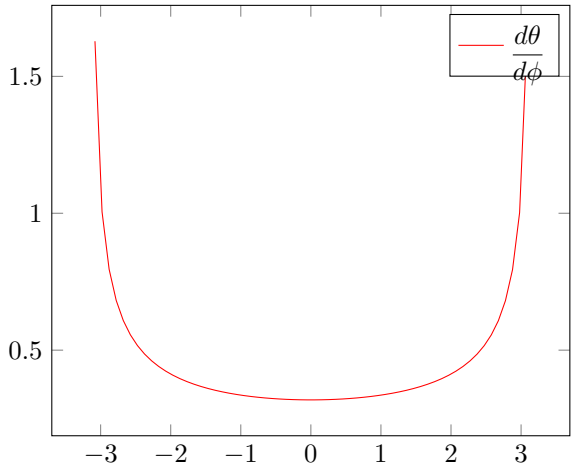
$$\theta > \arcsin \frac{10}{250} = 0.04rad$$

Thus, θ must be larger than 0.04 rad for it to be detected.

Now, the previous estimates of resolution assume a perfect signal with no noise. In reality, there will be a certain error δ on the value of the phase shift ϕ.

To evaluate the sensibility of our formula to a certain error δ, we must take the derivative.

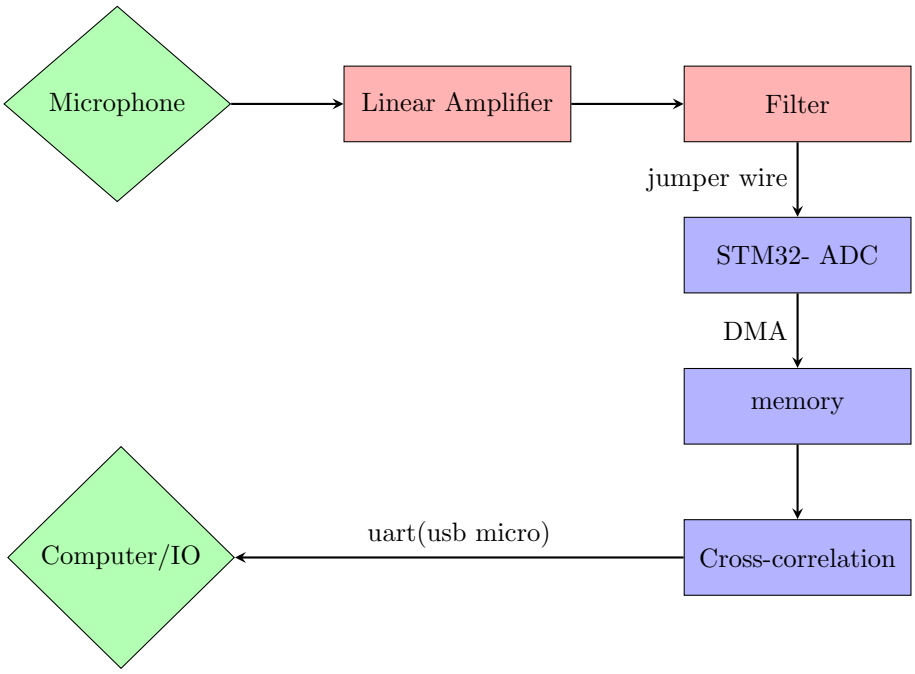
$$\frac{d\theta}{d\phi} = \frac{1}{\sqrt{\pi^2 - \phi^2}}$$



From this graph, we can clearly see that that our 2-element array is most sensible near $-\pi$ and π rad. Thus, near these two asymptotes any error will likely have a larger impact.

3.1 Block diagram

The following diagram illustrates a very simple pipeline that could be readily implemented with off-the-shelf components and a STM32 nucleo board.



3.2 Circuit diagram

In this section we will only discuss the implementation of an amplifier. I used a generic op-amp in a dip package. The op-amp was arranged in a simple voltage follower configuration. However, many other configurations could also be used. The whole circuit was implemented on a breadboard. At audio frequencies, the parasitic capacitance and inductance have little effect. Additionally, the op-amp and configuration we used had a linear phase response at the frequency of interest. Plus, the wires were short and of similar length. Therefore, we can reasonably assume the difference in phase between the two input signals was unaltered. Thus, a custom pcb was not necessary for the prototype. In short, little is required to make the circuit work.

3.3 Software

The previous description of cross-correlation probably left many questions unanswered. While there are many ways to implement the algorithm, the algorithm we present is a bit different, nevertheless it is easier to understand intuitively. Here is a simple set of steps that can be coded in any language.

We will assume both signals are stored in a finite list of S elements.

L: length of the signal

1. While (r < L)
   - While ($n < \frac{f_s}{2 * f}$),
     $p_n = signal_1[n] \cdot signal_2[n + r]$
     $n = n + 1$
   - $s_r = \sum_{i=0}^{n} p_n$
   - store s_r in a list (or array)
2. recover the index of the highest value in the list of s_r
3. $\phi = 2\pi - \frac{2\pi * Index}{\frac{f_s}{f}}$

As you can see, a crude but fully sufficient cross-correlation-like algorithm can be written in a few lines. Once θ has been recovered, all that is left is to plug the newly found value in the formula we previously derived : $\theta = \arcsin \frac{\phi}{\pi}$.

3.4 Discussion

The goal of this guide was to spark interest in the phased array while reducing complexity as much as possible. For instance, we purposefully avoided talking interpolation and other techniques to increase the resolution. We also, did not mention the filtering that goes before the cross-correlation. Many ways of filtering would have been adequate : FIR filters, FFT, Bessel filters, Chebyshev filters, SAW filters, or even a simple hardware RC bandpass filter. Filter design, be it hardware or software, deserves its own series of introduction projects.

If the given project were to be adapted to radio waves (UHF, SHF bands), a longer array would be necessary. In addition, software phase detection would be extremely hard, even with sub-sampler ADCs. Analog phase detection could, of course, be used. Any mixer could provide the phase difference between two signals, but the AD8302 would be one of the best fits. The circuit itself would have to be implemented on a pcb with lines of controlled impedance. Other issues such as feedback and mutual coupling would need to be watched for. The list goes on forever. The point is, this project is made orders of magnitude easier by targeting acoustic waves, but this isn't to say it is impossible to port the concept to radio waves. In fact, there are even COTS solution (see Kerberos and Krakensdr).

Finally, we must discuss the angle detection technique. What we presented in this document is formally known as TDOA (time difference of arrival). There are, however, many other positioning and ranging techniques. The most common are AOA (angle of arrival), RSSI (received signal strength), and TOA (time of arrival). These find many applications in various types of positioning technologies. UWB (ultra-wide band) positioning is a prime example.

4 Conclusion

In conclusion, we hope this short project helped you develop an interest in the world of phased arrays.