H4AR: Hearing for AR

Leon Chang
Electrical and Computer Engineering
Carnegie Mellon University
leonc@andrew.cmu.edu

Ramgopal Venkateswaran
Electrical and Computer Engineering
Carnegie Mellon University
ramgopav@andrew.cmu.edu

William Zhang
Electrical and Computer Engineering
Carnegie Mellon University
wz2@andrew.cmu.edu

***Abstract*—H4AR is a device for assisting hard-of-hearing people in their everyday lives, by visually alerting them about the direction of important sources of sound. Specifically, we will focus on alerting them to human conversation or voices around them.**

The device is a wearable pair of glasses, equipped with an array of microphones and simple augmented reality in the form of a small heads up display. This microphone array will identify human speech, and determine the direction of the sound relative to the individual. This information will then be displayed as a visual cue in the form of an arrow on the heads up display, pointing towards the direction of the source of sound. The aim of this is to increase the individual's peripheral awareness, allowing them to identify and turn towards people talking to them.

I. Introduction

The two ears on a human being offer a full range of sensory input where our other senses simply cannot. Our eyes have a peripheral vision range of only about $120°$ at any given point, and our sense of touch is only triggered at extremely close range. People hard of hearing simply lose this source of sensory input, and as such experience a much less complete awareness of their peripheral surroundings. It's already tough to communicate their handicap to others, it's even tougher when people attempt to talk to you from behind.

We propose our Hearing for AR (H4AR) device as a solution to this problem. H4AR is a wearable Augmented Reality device, equipped with an array of microphones that can identify sources of human speech, localize the speech, and display a visual cue of the direction of speech onto a glasses-mounted heads up display. H4AR greatly increases the peripheral awareness of those hard of hearing by translating the missing audio perception into a visual perception, thereby allowing their visual perception to have full peripheral awareness. The device is not only simple to use, with no direct human-to-device interaction required, but also it is also very portable, allowing users to wear it throughout the day.

II. Design Requirements

We have determined a set of requirements that our project must satisfy in order to meet its intended purpose. These include technical benchmarks - the product must determine the source of speech with good angular resolution, it must be able to isolate speech from noise, and it must do some with minimal response time - as well as usability constraints - the image on the heads-up display must be projected at a comfortable distance that the user's eye can focus on, the device must be easily wearable, portable and have good battery life. We will now discuss each requirement in further detail and specify the targets we want to meet for them.

A. Angular Resolution

Angular resolution refers to the precision with which the product can discern the direction from which sound originates. From our background research, we found that human hearing localization has at worst around $15°$ accuracy to the sides. Achieving this angular resolution would mean that we can supplement our users' hearing to an unimpaired level, and so we chose this as our target.

In addition to specifying the angular resolution, we also need to specify the range over which we aim to achieve this resolution. Since we aim to alert users to conversations outside their peripheral vision, and our users will already be visually aware of conversations in the range of about $180°$, our goal is to cover the remaining 180 (at the back of the head) that they might miss out on - i.e our product should cover the range beyond that of the human eye's visual span.

Based on the above information, our product needs to obtain a $\mathbf{15°}$ **angular resolution** over a $\mathbf{180°}$ **range**.

B. Response Time

The response time of our product is the delay between its receiving of audio input and its visual display of that input. This time delay will be affected by the rate at which the product can sample audio, as well as the time taken to evaluate whether a sample contain a human conversation, determine the direction of human conversations in that sample, and project this direction onto the heads up display. Our research informed us that that a natural human reaction time to aural stimuli is 0.17 seconds and to visual stimuli is 0.25, seconds which suggests that responding to visual stimuli is already slower than audio. We decided that our product should strive to have a delay on this order of magnitude: **at most $\mathbf{0.5}$ seconds**.

C. Noise Isolation and Conversation Identification

Noise isolation and human conversation recognition focuses on our product's ability to isolate and recognize certain conversation forms. Our product should be able to recognize human conversation at a noise level above 60dB as 60dB is the noise level of the typical human conversation. Furthermore, as average home noise is 40dB and 70dB is the average office noise, our product should also be able to **tolerate up to** 70**dB**

of non-human noise. Leveraging telephony's classification of voice frequencies, we concentrate on **human conversation that spans $300-3400$ Hz at a noise level of >60dB** [4], [5]. Due to the additional complexity and resources involved with discerning two separate speech conversations, we constrain the product to **only support** 1 **human sound source** - this is a reasonable constraint because only 1 person will be typically talking to the user at a given time.

D. Usability

The last three requirements are to do with the product's portability and physical usability. Noting that the human eye can only focus on virtual images >25cm away, we impose a simple requirement that the image of the sound source we project onto the heads up display **appears at least 25cm away** so that the human eye is able to focus on it. In addition, as the product is meant to be a wearable (i.e a headset), our target is 1**lbs** as popular market headsets are around this weight. Beyond visual focus and weight, our product also has a **battery life target of** 4 **hours continuous use** based on the premise of an average workday of working from $8-12$, recharging during lunch from $12-1$, and working from $1-5$ before going home at 5.

E. Design Requirement Summary

We conclude this section with a brief summary of the important metrics and our target objectives for the final product. The design objectives are summarized below:

- Angular Resolution: 15 degrees
- Response Time: 0.5 seconds
- Max Noise Level: 70 dB
- Min Human Conversation Noise Level: 60dB
- Number of human sound sources: 1
- Human Conversation Frequency: 300 - 3400 Hz
- Min Projection Distance: 25cm
- Max Product Weight: 1lb
- Min Battery Life; 4 hours

III. Design and System Description

We turn our focus to the specific design decisions that we've made and how they allow us to meet the overall product goal by satisfying the requirements outlined in the previous section. We give a fuller description of architectural and/or part design trade-offs, as well as other choices we had considered, in the section following this one.

A. Hardware Design

The system centers around a microphone array constructed on top of a headset. The microphone array comprises 4 digital **ST MP34DTO1-M** [6] omni-directional microphones, with an angular spacing of $\mathbf{60}^\circ$ between adjacent microphones (this corresponds to a straight-line distance of approximately **4 inches**). This provides us with approximately 300° of coverage centered on the back of the individual's head (definitely sufficient for our goal of 180°)..

The microphone array is integrated with the "Dragonfly" microcontroller unit (MCU) board via a **digital filter for sigma-delta modulator** (DFSDM). DFSDM has built-in sinc filters which can be used to eliminate certain frequency ranges (i.e noise). It also supports using dedicated DMA channels to write through to memory.

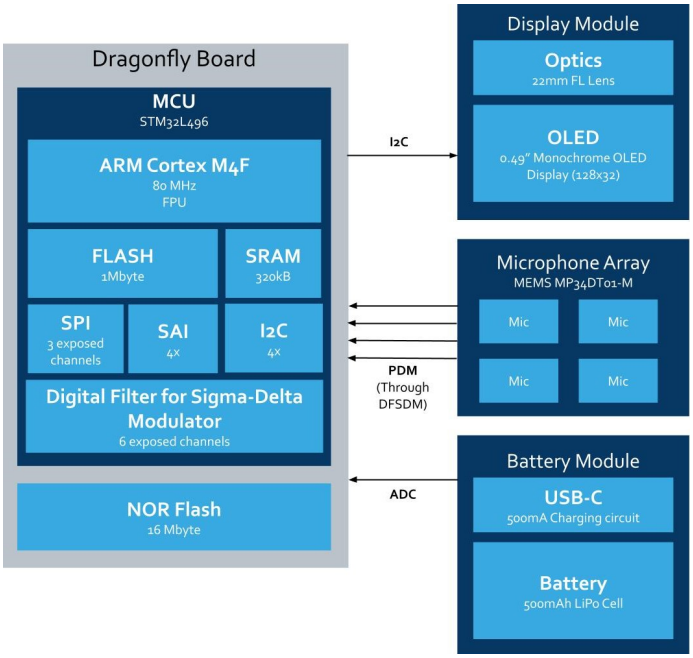


Fig. 1. HW Block Diagram

As seen in Fig. 1, the MCU we have chosen is a **"Dragonfly" STM32L496 development board**. The MCU is of the STM32L4 series which are designed primarily for low power consumption operations. It also comes in a small form factor which allows us to nicely integrate it into the headset along with reasonably strong hardware specifications: 1024 KB flash, 320 KB SRAM, 80 MHz clock, a floating point unit, and sufficiently exposed SAI and DFSDM lines to support 4 microphones.

Once the Dragonfly has determined the direction of the incoming audio source, it can utilize I2C protocol to tell the OLED screen what to project. The OLED screen is a **Micro SSD1306 IIC I2C OLED** which is a lower-power, small form factor screen with significant driver support. We chose a monochrome screen to save on power draw (since color is not a necessary part of our design). Furthermore, the OLED screen supports individual pixel lighting as compared to standard LCD backlighting; this is highly desirable as it allows the reflection of only the image on our projection surface.

The last major component is the battery source. Due to Dragonfly board restrictions imposed by the on-board voltage regulator, we are constrained to utilize a lithium polymer battery. We estimate the system current draw to be approximately 32.4 mA: the Dragonfly board draws around $\tilde{1}08\mu A$ / MHz when flash, SRAM, and all peripherals are enabled, the OLED board averages 20mA, and each mic averages 0.6mA. We utilize 42.4 mA to provide some slack with the estimation. We utilize a 3.7**V LiPo battery with** 500 **mAh capacity** as the

Dragonfly regulates input voltage to 3.3V and the regulator only supports 2.2V to 5.5V. With the capacity, we estimate around at least 13 hours of battery life. On integration of the hardware, we will take empirical current measurements to get a more accurate estimate of the battery life.

B. Algorithm Design

Our high-level goal is to estimate the direction of a sound source given a set of microphones positioned around the user's head. Given a pair of microphones lying on a straight line (assuming that we know which side of the line the sound source is from), it is possible to use the time difference of arrival between the signals at the two microphones to estimate the direction of the sound source.

Note that, with more microphones, this can be done with additional redundancy by computing the time difference between the signals at each microphone - the problem can be formulated as a simple linear system of equations and solving it gives us the desired direction. However, we also cannot have too many microphones - this is because we are limited by our hardware clock frequency (which limits how fast we can sample them, as well as how fast we can process the samples), by the number of ports available to us on our board (6 using the DFSDM), and by our power consumption. Therefore, it is desirable to minimize the amount of microphones to the extent that we stay within hardware limitations, and also give ourselves sufficient breathing room to perform the required computations on the sample.

Another factor to consider is that it would be most reliable to use time differences between adjacent pairs of microphones, because our computations assume that the microphones are "facing" the source of sound (the sound is not coming from the other side of the user's head), and that the sound travels directly to the microphones - if the user's head is in the way of the direct line between the sound source and one of the microphones, that would affect the accuracy of the computation. On the other hand, using adjacent microphones means that the spatial separation between them is lower, and our sampling rate needs to be high enough to detect time differences between microphones that are close together - we verified that we satisfy this requirement too (which we will explain in a later section).

A final, crucial factor that determines the algorithm we will use is the level of existing software support for it. As we will mention in a later section, there is a suitable library (AcousticSL) that we can use to obtain some rudimentary algorithms that identify from which direction a sound arrives using pairs of microphones. In order to minimize the amount of implementation work we will have to do, it is useful to go with a design that allows us to capitalize on existing libraries.

Based on the angular coverage requirement mentioned in a previous section, we chose a set-up with 4 microphones separated by 60 degrees. We then split these into adjacent pairs, one on each side. Taking the other factors outlined above into consideration, our algorithm was designed as follows: for every adjacent pair of microphones, we compute the possible angular directions of the sound based on this pair. For each pair, we will get two possible angles. By some geometric considerations, we can show that both pairs will theoretically agree (approximately) at one particular angle, and this would be the correct angle. So we choose this angle as a first approximation, and then further refine our estimate by choosing the microphone pair whose estimate is the most reliable out of the two based on which microphone pair is "facing" the approximate direction (that is, which direction is closest to being directly perpendicular to the line between the two microphones - the sound signal received by that microphone will be least affected by the user's head itself). At the end of this step of the algorithm, we will either get an angle estimate (or a placeholder value if there is no clear audio signal).

We also have one additional step, "windowing", that allows us to stabilize the output angle further and obtain a smooth signal, to control system and jitter noise. We split the overall set of possible angles into buckets, where each bucket represents some range of angles. Every time we get an output angle from the above part of the algorithm, we put it into the bucket coresponding to this angle. For every 20 such outputs (our window size), we find the bucket containing the most number of angles, and if it contains at least 10 angles (50% of the angles in our window size), we output it. Otherwise, we conclude that the signal is not reliable and we do not output a direction of human conversation.

This last step is a form of smoothing that prevents any flickering of the end output when it gets brief anomalous data, and our window size essentially controls how much we trade off latency for accuracy. The set-up with 4 microphones, where we only use adjacent pairs, might initially seem to offer minimal redundancy for a single output computation - however, by not using even more microphones, we are able to able to achieve low enough latencies that allow us to use the the windowing technique above to smooth the signal and this makes our system more robust.

C. Software Design

The first stage of the software design involves getting samples from the microphones at sufficient frequency, and doing some preliminary filtering on them. We can leverage the DFSDM to take care of the filtering and data conversion to PCM as we sample, as mentioned above. The DFSDM transfers the samples from each microphone into particular memory segments through DMA, and we can simply access them from there. Therefore, the sampling part of our software design is mostly painless, partly due to the hardware we selected.

The second and main stage of the software design involves computing the direction from which a human voice is arriving. We did this using a generalized cross-correlation algorithm (we will discuss it in more depth in the following paragraphs), using an implementation provided by the *osxAcousticSL* library, provided by ST. The library takes in a pair of microphones, and returns the angle offset of the sound source relative to the

microphones for a 1 ms sound signal. The overall flow is to call this library function for every pair of adjacent microphones to get the angle estimates, and then do the post-processing mentioned above to obtain the overall output.

The calculation of the direction involves two stages - the main first stage is to estimate the time difference between samples arriving at the two microphones, and the second is to use that information (and the known positions of the microphones) to estimate the direction. The signal processing in the first stage is where the bulk of the computation takes place.

To explain our decision to use the GCC-PhaT algorithm, we will briefly mention two other alternatives that we had originally considered, and which are also offered by the *AcousticSL* library. In all three cases, it works with samples of 1 ms (note that we need to verify that samples of 1 ms are sufficient to distinguish at the desired angular resolution - we will verify this in a later section). The first choice is to run a simple cross-correlation (XCORR) between them. The second choice is to perform a generalized cross-correlation, known as GCC-PhaT (generalized cross correlation - phase transform) - this does some extra weighting of different frequencies, to make it more robust to reverberations and noise. The third choice is a lesser known algorithm called the BMPH algorithm. We found that GCC-PhaT has been fairly popular in human sound localization research, and that it generally gives significantly better performance over regular cross-correlation in noisy settings. Therefore, this is our preferred method.

One risk in using GCC-PhaT was that it is a computationally expensive algorithm relative to simple cross-correlation. While simple cross-correlation can be implemented in both the time and frequency domains, GCC-PhaT is usually implemented in the frequency domain (including this library's implementation). This involves doing two Fourier transforms (FTs) per the microphone sample - one to get it into the frequency domain, and one inverse transform after re-weighting frequencies to get it back into the time domain. Since FTs are relatively expensive, especially in a real-time setting, this operation forms the bulk of the computation.

By cutting down other latencies in our system, such as the latencies associated with displaying the direction on the screen and with sampling the microphones, our device is able to run this relatively powerful algorithm while staying under the latency requirement (as we will show in a later section). Additionally, using just 4 microphones also helped us save on computation here.

D. Communication Design

During our design process, we worked to minimize the complexity of the overall system, removing modules and components and opting for the simplest solution. In doing so, we have our communication interfaces at a minimum. Table I provides a summary of the communication interfaces as well as protocols we are using.

1) Microphones: The majority of communication required for our product solution to function is between the microphones and the MCU. Our microphones of choice are digital microphones, meaning that they have an amplifier as well as an ADC built into their package. The signal outputted by these microphones are in a Pulse Density Modulation (PDM) format, which is conceptually similar to Pulse Width Modulation (PWM). These digital microphones take a clock signal which defines the sampling rate as well as the transmission rate, while outputting a single bit binary stream, see Fig. 2, which can then further be converted into Pulse Code Modulation (PCM), see Fig. 3 for use by the CPU.

TABLE I
COMMUNICATION SUMMARY

Component	Protocol	*Hardware Interface*	*# I/O*
4x Microphones	PDM[a]	DFSDM	3
OLED Display	I^2C	I^2C	2
Battery	N/A	ADCC	1

[a]Converted to PCM

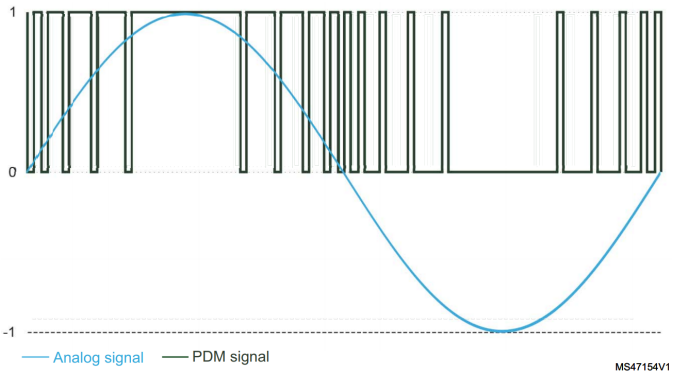


Fig. 2. PDM Signal [1]

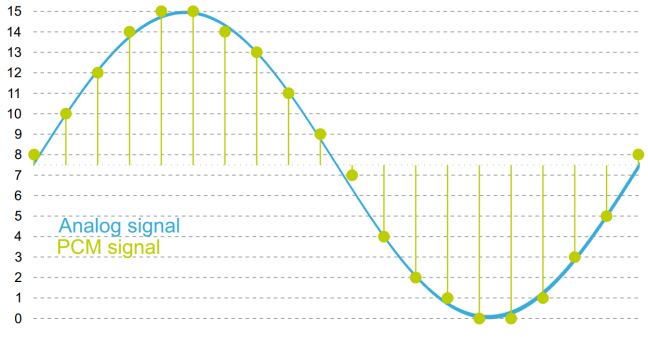


Fig. 3. PCM Signal [1]

Since we are targeting speech applications, we will be driving these microphones with a clock frequency between 1 and 1.5MHz as per the datasheet. These microphones also support a "left" and "right" channel selection which controls whether the microphone sends data on the rising or falling edge of the clock. This allows us to have the data of two microphones on a single data line. We interface these PDM microphones with the MCU through the DFSDM digital peripheral. For our

target 4 microphones, we interface through the DFSDM with a minimal amount of I/O: two data lines and one clock line.

As in Fig. 4, the DFSDM will provide the MCU with filtered samples in Pulse Code Modulation (PCM) format. Samples are transferred internally through DMA to a system RAM buffer, with a separate DMA channel per microphone. This data can then be used by the CPU for the TDOA algorithm.

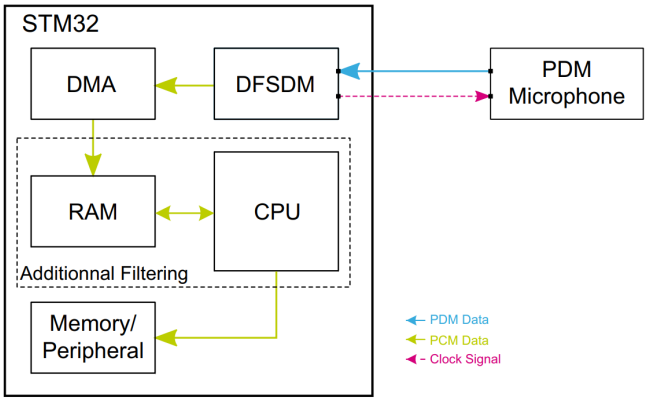


Fig. 4. Microphone data acquisition for a single mic. using DFSDM [1]

2) OLED Display: Our product solution also communicates with the OLED display to portray information to the user. This communication is achieved through the I2C protocol, requiring only one data line and one clock line for a total of two I/Os. Our production display has a resolution of 128x32, while our development display has a resolution of 128x64, as such, to write out to the display, the image must first be buffered into the RAM (512 bytes for the production display, 1 Kilobyte for the development display). The image is then written out to the display via I2C. The SSD1306 chip can be driven with different I2C clock rates. In the interest of balancing our latency requirement of 500ms, as well as saving battery power, we drove our I2C interface with a clock rate of 400kHz (fast mode), giving us a 23+ms delay. To further bolster the speed, we used techniques that reduce the amount of data sent over the wire for each frame, as we are not refreshing the whole screen every time - we only update the screen when absolutely necessary to avoid extra overhead.

3) Battery Module: The final communication interface is in the battery module. In order for the user to know the battery level and know when to recharge the battery, we will need to read information from the battery. This is a simple communication, achieved using an one of the onboard ADC channels with one I/O used. The ADC will be used to intermittently read the voltage of the battery, and calculate the battery percentage from the voltage, alerting the user when a recharge is needed.

E. Optical Design

In addition to the software and hardware aspects of our project, we also need to use some optics to project the image formed on the OLED screen (likely an arrow indicating the direction of sound) onto a small plexiglass reflector that will be positioned in front of the user's glasses. The set-up for this can be seen in Fig 5. A double convex lens first forms a virtual image of the OLED screen, on the same side of the lens as the OLED screen. The plexiglass reflector then catches this virtual image, so the image is seen on the reflector.

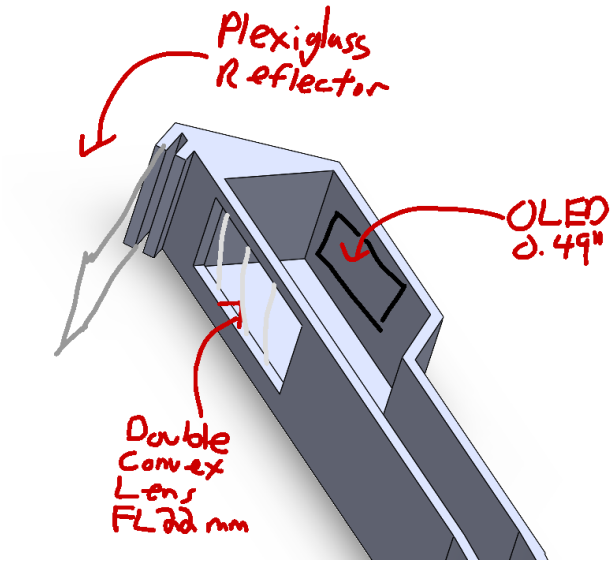


Fig. 5. Optics System Setup

The reason for this is that the eye needs 25cm+ of "distance" to focus on an image - by choosing a lens of appropriate focal length, we are able to achieve this. Our optical design is novel in that it does not utilize any mirrors, unlike existing designs - the reason for this is that we were unsatisfied with such designs as they create a lot of additional bulk. Our setup is a lot more compact, requiring just under an inch of clearance. This requires slightly thicker and heavier lenses than most designs - we bought a few different lenses with different focal lengths, and found that both theoretically and experimentally, the one with $22mm$ was optimal.

F. Physical Design

The product solution has to be portable and wearable on the user's head. To get to that goal, we have designed a device that is meant to be attached to a pre-existing pair of glasses, be it prescription or fake lens glasses. The device is mounted on the two legs of a pair of glasses, with a band that wraps around the back, see Fig. 6. It is designed to be removable in order to charge without disrupting the normal use of the user's eyeglasses. An image of the device is shown in Fig. 7. Note that, for increased stability and comfort to the user, we also added some foam padding and an elastic strap that goes around the user's head.

G. Overall Feasibility of Our Solution

We now present some approximate calculations that explain mathematically why our physical positioning of the microphones, the hardware we have chosen, and our software solution were sufficient to achieve our design requirement

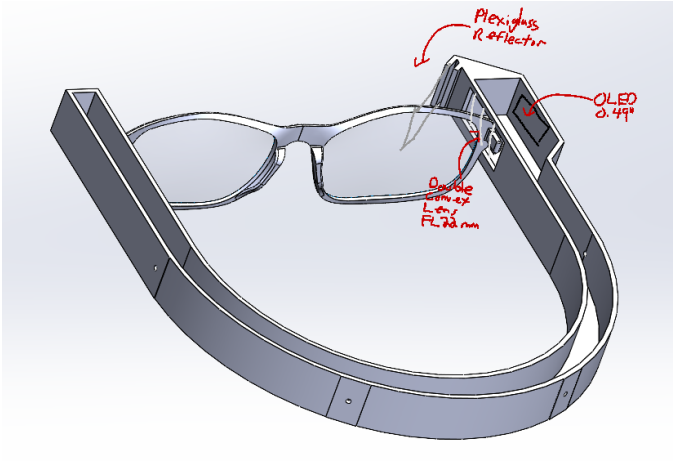


Fig. 6. Design mounted on glasses

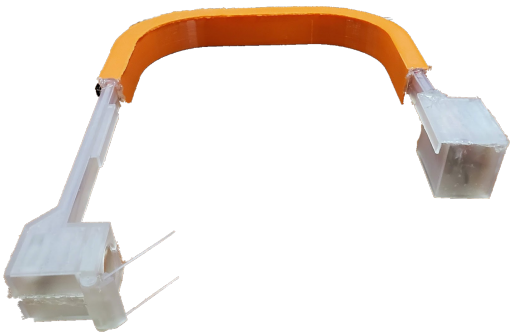


Fig. 7. Top-down view of the device

of 15 degrees angular resolution with at most 500 ms lag response time. Note that these calculations were done at the design stage, and so they explain why the design is feasible in principle - we will turn to the actual measurements we took and the results we obtained in a subsequent section.

First, we note that working with 1 ms samples is sufficient to distinguish two sound sources. For this, we need to ensure that there is a good amount of overlap between the signals received by adjacent microphones. This is important to calculate the time difference between them - for instance, if we had completely non-overlapping signals, we would not be able to tell the difference between them. Since the microphones are 4 inches apart, the time difference between the signal reaching the two microphones is at most $(4 \text{ inches})/(300ms^{-1}) = 0.3ms$. Therefore, we will have at least 0.7 ms of overlap, which allows our cross-correlation based algorithms to function as desired.

Next, we need to check that our sampling frequency is high enough to meet the 15 degree angular resolution requirement. Given an angular displacement of θ from the center line between the 2 microphones, we have the following, where c is the speed of sound:

$$\Delta\text{distance to mic} = (\text{distance between mics}) \cdot \sin\theta \quad (1)$$

$$sin\theta = \frac{4 \text{ inches}}{c \cdot (\text{time difference})} \quad (2)$$

As a conservative estimate, say we wanted a time difference corresponding to at least 10 samples between two angles that are $15°$ apart (our angular resolution). Using our formula above, that means we would want, for all θ in the range $[0°, 75°]$:

$$\frac{4 \text{ inches}}{c}\left(\frac{1}{\sin\theta} - \frac{1}{\sin(\theta + 15°)}\right) \geq 10 \cdot T_{sampling} \quad (3)$$

Note that the range is restricted from $0°$ to $75°$ because we're assuming that the microphones are facing the sound source (which is the only case in which we will use this estimate). The LHS is minimized at $\theta = 75°$. Then we have that $T_{sampling} \leq 2.31 \cdot 10^{-7}s$, and so our sampling frequency from each microphone is at least 0.433 MHz. We have 4 microphones, so our overall sampling frequency would just need to be 1.73 MHz. Both of these figures are fine - we can sample from the individual microphones at between 1 to 1.5 MHz, and our board has a clock frequency of 80 MHz, both of which exceed these respective figures.

Note also that this was in fact a very conservative calculation. We assumed that we needed at least 10 samples spacing between theta that are 15 degrees apart (in fact, it could even be possible to do it with less than 1, because we just need to be able to notice some difference in the signals over the course of 1 ms worth of samples). We also assumed a worst case scenario of the sample coming from an angle of 75 degrees to the center between the microphones - our physical design ensures that there will always be some pair of adjacent microphones such that this angle is at most 60 degrees.

The library documentation also gives its own formula for the minimum sampling frequency (under different assumptions, and using a different method) which comes out to just 18 kHz, and our sampling frequency far exceeds this amount as well. Therefore, we were confident that our design would be able to theoretically achieve the desired angular resolution, and this was indeed true.

Finally, we verified before implementation that our design would likely meet the 500 ms response time requirement. We did this by getting an order-of-magnitude estimate of the number of clock cycles needed for our processing of each sample. Even though the library doesn't provide these details pertaining to the runtime of the GCC-PhaT algorithm, one of the papers mentioned we looked at [9] gives the number of clock cycles used in its frequency domain implementation of GCC-PhaT - we can use this as an (extremely rough) estimate. According to the paper, performing GCC-PhaT on two data blocks of size $N = 256$ takes $2.1 * 10^6$ clock cycles. If we sample at 500 kHz, since the library uses 1 ms blocks, we would be working on blocks of size $N = 500$. The complexity

of a Fast Fourier Transform is O(NlogN) so using this scaling, we get approximately $4.8 * 10^6$ clock cycles (N increases by a factor of 2, logN from roughly 8 to 9). We do this for 2 pairs of mics, giving $9.6 * 10^6$ clock cycles. This is the most expensive computation we will perform - to be conservative, let's double our processing time to account for other computations. Then we take $19.2 * 10^6$ clock cycles total in our processing, which is still under 500 ms, our target.

A risk we had planned for was that, while this was still below our target of 500 ms response, it was fairly tight - especially given the fact that the internal library's implementation of GCC-PhaT is a black box. However, by cutting down on other latencies, we were eventually able to meet this comfortably.

IV. Design Tradeoffs and Trade Studies

A. Architectural

The product's initial proposal was based on the architecture presented in a Euronoise 2018 paper on sound source localization in 360 degrees using a circular microphone array [8]. The architecture presented in the paper utilizes a single board RIO that combines a FPGA and a real-time processor for fast data acquisition and calculations. Our proposal also considered the architecture of several other papers that utilize beamforming techniques and a single microcontroller unit (MCU). We ultimately decided to go with utilizing only a MCU due to a mix of various competing factors. Although a FPGA theoretically could provide faster sampling than a MCU, the sampling rate of the on-board DFSDM is sufficient. Furthermore, a FPGA would require additional power consumption and also has additional hardware constraints that we deemed to be burdensome: we would have to either use a SoC FPGA which has a large form factor or manually handle interfacing a FPGA with a MCU through communication protocols like SPI which has unnecessary engineering overhead.

B. MCU Evaluation

Due to the hardware constraints imposed by a microphone array on the board, our process first evaluated the right MCU to use. Our official requirements stipulate that the MCU must be able to support at least 4 microphones and up to 8 microphones along with being in board form. Portability was also a significant consideration; many reference and development boards exist that give us more than enough I/O, but are much too large to be integrated in our product solution. Our board evaluation primarily focused on STM32L4 and STM32F4 boards that could be acquired commercially. We decided on the Dragonfly STM32L496 Development Board after discarding several alternative boards. First the STM32L4 series were designed for low-power consumption in comparison to STM32F4 boards which would enable a longer battery life. Although the Dragonfly only has a 80MHz clock, boards with faster clocks do not have sufficient microphone interfaces (i.e DFSDM, SAI) to be considered viable. Furthermore, the Dragonfly board is optimized for portability with a tiny form factor, measuring in at only 0.7" x 1.4", as well as very competitive flash and SRAM sizes in comparison to other boards. As such, although we are trading the clock frequency and additional processing capabilities, we argue that the Dragonfly provides the best risk mitigation with on-board support of up to 8 microphones and is also designed for low-power scenarios.

C. Microphone Array

For constructing the microphone array, the first design decision we made was whether to use analog or digital microphones. We committed to digital microphones as analog microphones have additional overhead since they would require supplementary A/D converters and pins on the MCU board to interface correctly. The particular digital microphone we ended up selecting was the ST MP34DT01-M. One of the main benefits to this particular microphone is that it has a known product viability (as part of the Bluecoin device) and interface documentation with the STM32L4 low-power board series. Furthermore, this microphone unlike other considered microphones is omnidirectional and board-mounted with a comparable signal to noise ratio of 61 dB and -26 dbFS sensitivity and an acoustic overload point of 120dbSPL.

D. OLED Screen

Another area of trade-offs we made was in determining the OLED screen. Our primary constraints motivating this component focused around dimensions and the existence of libraries. Considering the possibilities, we finalized on the Micro SSD1306 IIC I2C OLED with one of its main selling features being that the screen is only 0.49" as opposed to common designs of 0.9". Furthermore, the SSD1306 has many libraries available and we deem I2C to be adequate enough for a low resolution screen. An auxiliary benefit derived from this device is power consumption benefits: the monochrome device saves power in comparison to color screens and does not rely on a backlight.

E. Battery

There were not many significant trade-offs made for the battery. As mentioned previously, the battery voltage is constrained by the requirements of the board. As the Dragonfly has a NCV8170 LDO regulator, the input voltage requirements are constrained from 2.2 V to 5.5 V which makes the common 3.7 V a de facto option. When it came to evaluating capacity, we chose the 500 mAh as a compromise between price points and an estimated use-time exceeding at least 8 hours which can be satisfied with the 500 mAh capacity assuming the current draw remains on average below our estimated 42.4 mA.

F. Software

As discussed above, the significant bulk of the software implementation involves implementing and calculating the time difference of arrival based on the input microphone streams. Our design relies on osxAcousticSL / X-CUBE-MEMSMIC1 libraries which provide implementations of the XCORR algorithm and GCC-PhaT under certain constraints (i.e number of microphones, arrangement of microphones,

spacing between microphones). The infrastructure provided by X-CUBE-MEMSMIC1 also contains a hardware abstraction layer that supports STM32L4 which serves as a convenient bootstrap.

Although the ST libraries do have constraints that are not entirely compatible with our product formulation, (e.g. ST requires that for a four microphone configuration, the microphone pairs must be perpendicular, which would not work for us because the user's head is in the way). However, as implementing algorithms like GCC-PhaT has innate complexity, utilizing osxAcousticSL / X-CUBE-MEMSMIC1 still made sense, since the libraries are developed with integration of external microphones and within the ST ecosystem as a whole.

An important software trade-off, also mentioned earlier, is windowing. We are trading off latency for increased accuracy. Having cut down other sources of latencies to be able to meet the latency benchmark comfortably, windowing allowed us to get a smoother signal and improved accuracy with increased latency that would still meet the benchmark.

V. Implementation Procedure

A. Hardware Implementation

The hardware implementation plan was characterized by a modular design that was implemented in stages by priority. The design is made up of three main hardware components: (1) microphone array, (2) display module, and finally (3) battery module. The microphone array, being the highest priority, was implemented first. This priority was set since the software implementation requires the microphone array to proceed. The microphone array was first tested in its predetermined configuration before being mounted onto our design. This allowed us to verify the validity of the hardware while we constructing and implement the microphone module.

The display module implementation efforts followed after the implementation of the microphone array, as it carries the second highest priority. The battery module carried the lowest priority, as it is only important when we are undergoing integration. As such it was the last module that we implemented. The completion of the battery module marked the beginning of the final hardware implementation stage: integration. The integration stage involved the three hardware components into a portable package.

B. Software Implementation

The software implementation plan was coupled tightly with the hardware implementation plan, with multiple stages that are executed parallel to the hardware implementation stages to the highest extent possible. The first stage was bootstrapping the MCU. This stage involved setting up the firmware, deciding on abstraction layers, setting bits in the MCU to enable peripherals as well as setting up the shared toolchain. After the completion of this stage and the completion of the microphone array in hardware, we moved on to the second stage: interfacing with the microphone array. This stage involves interfacing with the DFSDM or SPI peripheral in order to receive microphone readings, as well as manipulating the data retrieved to be useful. The third stage was using the microphone array data to implement the localization algorithm, using generalized cross correlation (GCC-PhaT). We then spent time on post-processing and fine-tuning the parameters of the system. After localization, we interfaced with the I2C OLED display to display date in stage four, and finally interface with the battery to get battery level readings during stage five.

VI. Metrics and Validation

This section focuses on the validation of the product. In this section, each requirement is re-iterated from the design requirements section and our validation strategy and obtained results, are discussed afterwards.

A. Angular Resolution Validation

A significant goal of our product is to satisfy the angular resolution requirement of $15°$. To verify that our product does indeed have such a resolution, we placed the device on a table in a noise-controlled room. One of us then walked around the device, and at $5°$ intervals, stopped and talked at the device. We focus on the rear side of human vision, i.e the $180°$ outside of the range of human peripheral vision. We retrieved the angle computed by our system and checked how it matched up with the actual angle. From our results, based on 193 samples, we were within $\pm 15°$ relative to the true angle 100% of the time, while we were within $\pm 7.5°$ relative to the true angle 94.8% of the time. Therefore, we achieved our angular resolution benchmark. The graph of obtained angles versus actual angles is show in Fig. 8.

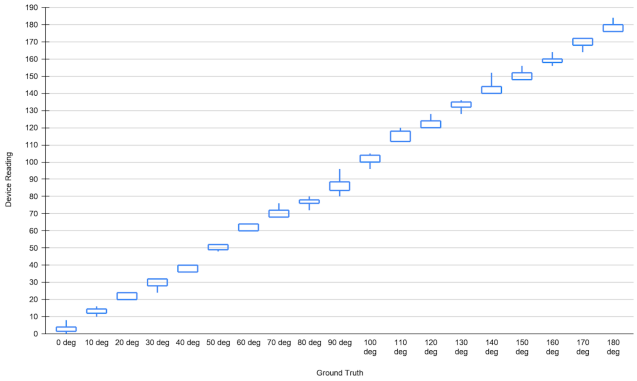


Fig. 8. Obtained angles vs actual angles; the box represents the 1st to 3rd quartiles of obtained angles

B. Response Time Validation

The next metric focuses on response time. The target response time requirement was 0.5 seconds for the system. In particular, we measured the time from detection to display - from when a person would speak, to when the corresponding arrow would be shown on the screen. The set-up had a person speak from behind (at varying distances and angles);

the wearer would start a clock on hearing speech and stop the clock on seeing the arrow. Based on 15 samples where the speaker is 6ft behind, we had a mean latency of 355ms with standard deviation of 124ms. While this measurement might seem to depend on the wearer's reaction time, note that both starting and stopping the clock has this associated reaction time delay and, in fact, stopping the clock would have higher delay since human visual reaction time is greater than audio reaction time; therefore, this test is actually slightly conservative and passing this test allows us to verify that we are definitely hitting the latency benchmark.

C. Noise and Human Conversation Recognition Validation

A followup requirement is handling human conversation ranging from 300 - 3400 Hz at a voice level of $>$ 60dB with up to 70dB of non human noise. To verify our product's compliance with this requirement, we evaluated the product in four noise situations: (1) a very quiet room; (2) an "office-like" environment with AC/fan and keyboard noise; and (3) a "very-noisy" restaurant environment. In all cases, we verified that if an individual is speaking, our system is able to identify such a fact and that the identified signal is within the angular resolution bounds. However, in cases 2 and 3, when no individual is speaking, the product sometimes registers the sound and outputs a direction - while this is usually just some brief flickering and does not impede the general use case, this represents a potential area for improvement.

D. Usability Validation

The final three requirements focus on usability which will be discussed together: $>$ 25cm projection, 1lbs weight, 4 hours battery life, and portability. For the ¿25cm projection, we visually verified that the image was clear and compared the apparent distance to known objects as a reference. To validate portability, we weighed the product - it is only 5 oz, and very lightweight. Finally, to validate the battery life, we left the device on with a continuous sound input. We stopped the test after 12 hours, when it was still running. Analyzing the current, the peak draw is 17.8 mA with a 28 hours theoretical usage, which is more than sufficient. We also conducted a few additional tests. We checked that the product still works correctly and stably even when the wearer is (1) walking around on a flat level; (2) walking upstairs and downstairs; and (3) crouching. We also did a "head tilt test", where the user tilted his head 0 to 45 degrees to verify device secureness and also correctness.

VII. Project Management

A. Schedule

We were on schedule for the most part, though we experienced a few delays along the way early on in the project - a delay in the BOM creation led to some part delays, however due to our previously built in slack time for part shipment of two weeks, we were not too behind and we eventually caught up. Towards the end, we did a software rewrite for windowing, and integration took longer than expected (so we pushed it back to phase 4). Since we had enough built-in slack time, we accomplished the project successfully on time. Our Gantt chart is attached in appendix B.

B. Team Member Responsibilities

Responsibilities were assigned as follows:

1) Leon: Physical Design, Hardware Implementation and Integration

This involves the CAD design and fabrication of all components of the system, as well as the wiring, testing and implementation of hardware components. Some software tasks will also be part of Leon's responsibilities, namely the display driver, as well as working on setting up the firmware with Will.

2) Ram: Signal Processing, Algorithm Development

This involves setting up the DFSDM peripheral to apply digital sinc filter(s) to the audio input, as well as potential further processing on the CPU. Algorithm development involves research and implementation of the time difference calculation algorithms, as well as iterating on the algorithm to make it more efficient, in terms of speed and/or battery usage.

3) Will: Algorithm Development, Software Implementation

Algorithm development involves working with Ram to develop the time difference calculation algorithms, as well as doing optimizations in parallel. Software implementation includes setup of our MCU, flashing it with our custom firmware that enables the peripheral we need to use. It also includes the implementation of code to read battery levels, as well as code to integrate all aspects of our software design together.

C. Budget

We have included in Appendix C our Bill of Materials, containing the parts we bought. We ended up using all of the parts, except for the 45mm and 106mm lenses. We bought extras of some parts, like the microphones, but most of these extras were used in the course of development as well (so that all of us could work parallely).

D. Risk Management

Most of the risk in our project was during the implementation stage, since there was uncertainty associated with the accuracy and latency of our algorithm.

To mitigate the risk associated with accuracy, we planned to use up to 8 microphones for increased redundancy, and selected a board with the capability to support these many microphones. We chose the Dragonfly because it provided the best risk mitigation with on-board support of up to 8 microphones - particularly, in the possible eventuality that we would have required 8 microphones,instead of using the DFSDM (which only supports up to 6 microphones), we would have used the serial audio interface (SAI) to interface the microphone array with the Dragonfly board. SAI supports up to 8 microphones (four pairs of mics where for each pair, one is rising-edge triggered and the other is falling-edge triggered). However, SAI would require additional MCU support to de-interleave the data stream from pairs of microphones and also

filtering; this fact, along with the filtering capabilities of the DFSDM, made it our first choice [7].

Thankfully, the accuracy did not end up being an issue - we achieved sufficient accuracy using 4 microphones coupled with some post-processing techniques like windowing.

The other significant risk was the latency of the system - because the details of AcousticSL's internal implementation of the GCC-PhaT algorithm are not public, we relied on the latencies reported by papers using this algorithm to arrive at our estimate and conclude that the design would work. We also prepared ourselves for the case where, if it was too slow, we would implement GCC-PhaT ourselves; again, we were careful in choosing our board to ensure that it had good support for digital signal processing (DSP), and that there were existing libraries that we could use with it that contained important primitives (like bandpass filtering, Fourier transforms, etc.)

Finally, there was some additional risk to do with scheduling, in particular in terms of our access to Techspark resources and being able to iterate on our physical design. We negotiated these risks by getting an early prototype out very early in the implementation stage, as seen in our Gantt chart; this insulated us from having to worry too much about this issue, especially since our early prototype was designed very flexibly to support anywhere from 4 to 8 microphones - so even if some parts of our software design would have changed, we would still have been able to use the same physical design. This early prototyping also turned out to be very useful from the standpoint of tuning the parameters of our device and ensuring our implementation works well during development.

VIII. Related Work

Our product is in a unique space - we have not come across other devices that attempt to perform human voice localization with an AR display in a portable form factor. In terms of sound localization functionality alone, one existing device on the market with similar capabilities is Bluecoin [10]. Note, however, that we did not use this because it also contains several other functionalities that are not necessary to us, allowing for less fine-grained control and giving extra overhead, potentially compromising our latency requirement. In the research space, there has also been a lot of interest in developing techniques devoted to accurate human sound localization in the face of noisy environments - for examples, see [8] and [9]. These are not as constrained in terms of portability and form factor as ours, and are more focused on optimizing for accuracy as far as possible, with minimal latency.

IX. Summary

Our system was able to meet almost all of our design specifications, apart from one set of tests where we wanted it to be completely able to ignore all noises even in very noisy environments when there is no human sound altogether - as mentioned earlier, note that this does not really show up in general use (in particular, we still can recognize human speech and focus on that in the presence of background noise).

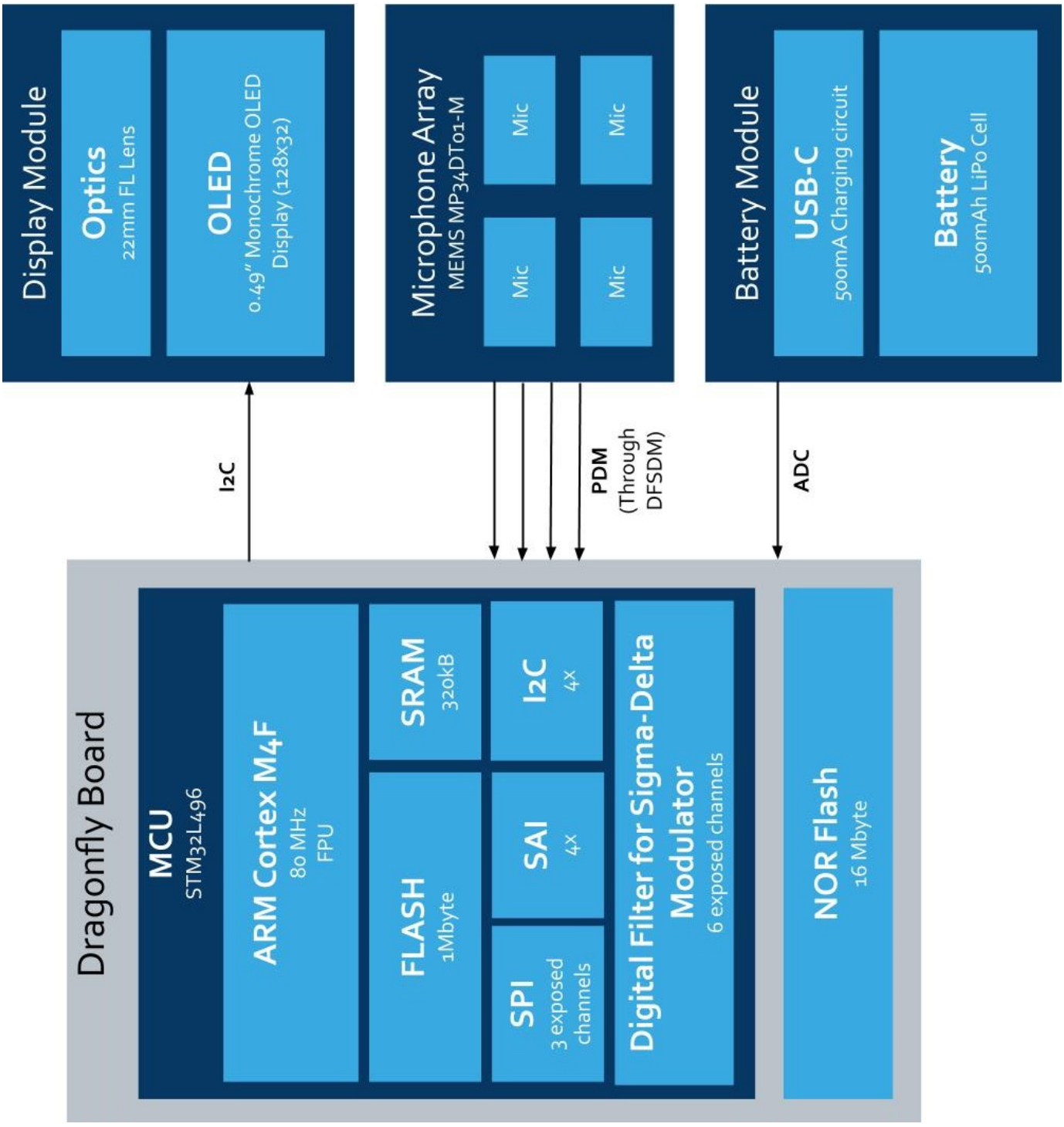
Two things that worked in our favor were that power consumption was not as issue, and that we could achieve lower latencies than expected by eliminating a lot of overhead when sampling from the microphone and displaying the visual cues to the user. This enabled us to focus on accuracy. This also leads us to some areas for improvement include potentially performing some extra filtering, in order to perform better in noisy environments, and perhaps even looking at more sophisticated techniques like machine learning to focus on human sound recognition. We know that because our latency and power consumption are low enough, these would be interesting and feasible avenues to explore, to improve our accuracy even further. Other extension areas include distinguishing between different classes of useful sound (e.g. vehicular/traffic sounds), and being able to identify multiple sources of human sound at the same time.

The main lesson learned was that planning out our design in detail before the implementation was exceedingly useful - even though there were still some surprises that emerged during implementation, the fact that we had a good working knowledge of the system and what we wanted to do before we got our parts enabled our development to proceed very smoothly. Another helpful lesson was that we streamlined the project early on, to focus on human voices and particularly the case of one human voice - this was helpful, because from then on, we could set concrete metrics and our device could focus on doing just a few things, but doing them well.

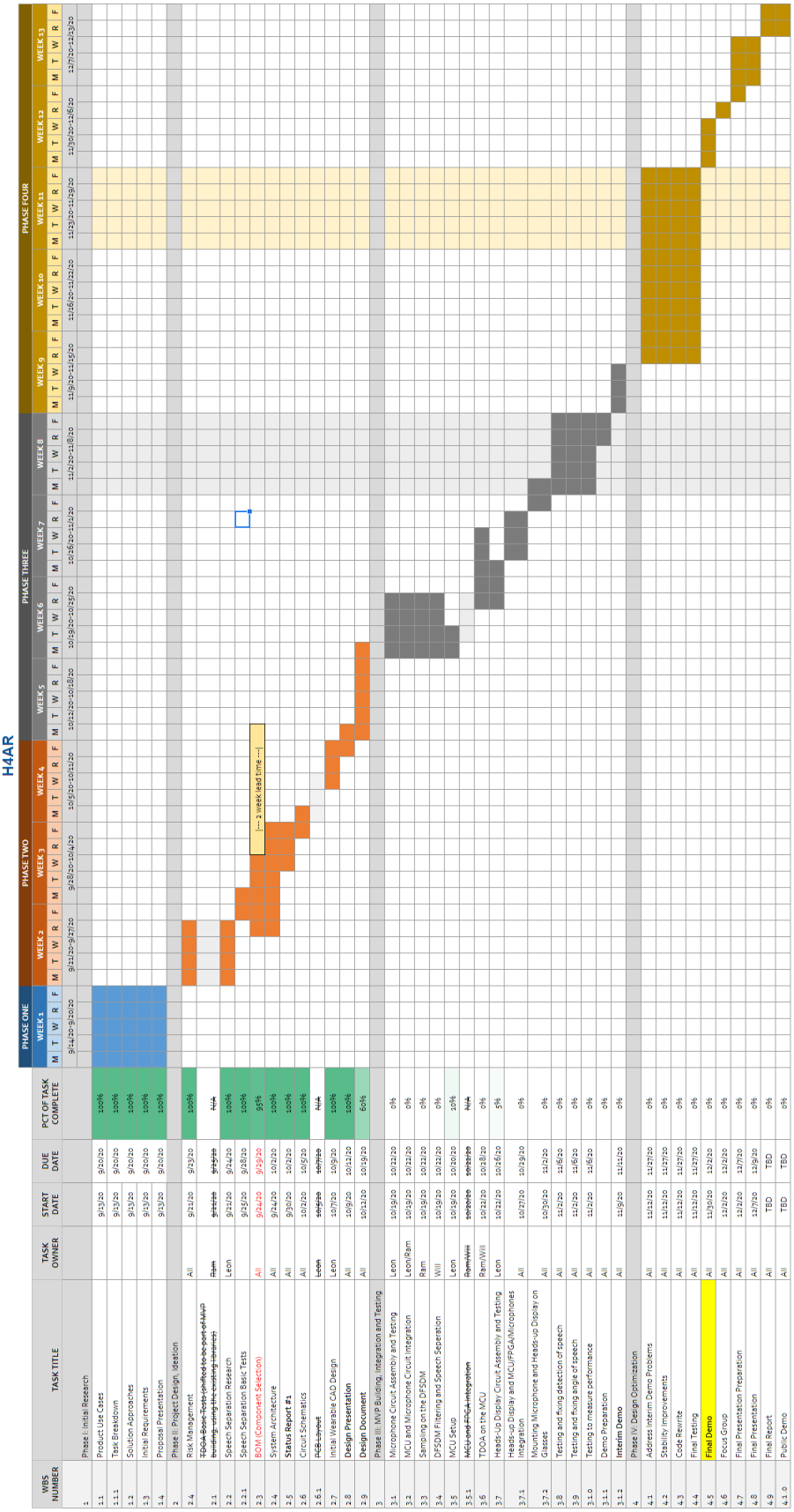
References

[1] "AN 5027 Application note Interfacing PDM digital microphones using STM 32 32-bit Arm Cortex MCUs", July 2019.
[2] "https://en.wikipedia.org/wiki/Vision_span"
[3] "Passive Eye Monitoring, Algorithms, Applications and Experiments"
[4] "https://myhealth.alberta.ca/Health/Pages/conditions.aspx?hwid=tf4173"
[5] "https://www.its.bldrdoc.gov/fs-1037/dir-039/_5829.htm"
[6] "https://cdn-learn.adafruit.com/assets/assets/000/049/977/original/MP34DT01-M.pdf"
[7] "https://www.st.com/resource/en/application_note/dm00380469-interfacing-pdm-digital-microphones-using-stm32-mcus-and-mpus-stmicroelectronics.pdf"
[8] "http://www.euronoise2018.eu/docs/papers/432_Euronoise2018.pdf"
[9] "https://ieeexplore.ieee.org/document/6532229"
[10] "https://www.st.com/en/evaluation-tools/steval-bcnkt01v1.html"

APPENDIX A
ARCHITECTURE BLOCK DIAGRAM



Appendix B
Gantt Planning Chart



Appendix C
Bill of Materials

Item #	Part Name	Quantity	Manufacturer	Purchase Link	Comments
1	ST-LINK	3		https://www.amazon.com/DAOKI-ST-Link-	For programming the board
2	Dragonfly STM32L496 Development Board	3	Tlera Corp	https://www.tindie.com/products/tleracorp	Small, low-power development board with STM32L496 MCU on it, necessary interfaces exposed
3	Adafruit PDM Microphone Breakout with JST SH Connector	16	ST Microelectronics	https://www.adafruit.com/product/4346	PDM Microphone breakout boards for Microphone array
4	Adafruit Lithium Ion Polymer Battery - 3.7v 500mAh	3	Adafruit	https://www.adafruit.com/product/1578	LIPO Battery
5	Adafruit Micro-Lipo Charger for LiPoly Batt with USB Type C Jack	1	Adafruit	https://www.adafruit.com/product/4410	LIPO Battery Charger
6	Adafruit Monochrome 0.96" 128x64 OLED Graphic Display	1	Adafruit	https://www.adafruit.com/product/326	OLED Display, larger for prototyping
7	0.49 Inch Micro SSD1306 I2C OLED Display	1	Walfront	https://www.amazon.com/SSD1306-Displa	Tiny OLED Display, for final product
8	Adafruit JST SH 4-pin Cable	18	Adafruit	https://www.adafruit.com/product/4397	Cable for I2C interface
9	1" Square Mirrors	1	N/A	https://www.amazon.com/Craft-Mirrors-12	Mirrors for reflecting OLED image
10.1	45mm Focal Length Lenses	1	N/A	https://www.amazon.com/Blisstime-Bi-cor	Lens for virtual image projection
10.2	26 x 22mm FL, Grade 1, Double-Convex Lens	2	Edmund Optics	https://www.edmundoptics.com/p/26-x-22	Lens for virtual image projection
10.3	52 x 106mm FL, Grade 1, Double-Convex Lens	2	Edmund Optics	https://www.edmundoptics.com/p/52-x-10	Lens for virtual image projection
11	22 AWG Hook Up Wire	1	N/A	https://www.amazon.com/StrivedayTM-Sc	Wires
12	CD Cases	1	Staples	https://www.staples.com/Staples-Standar	Plastic used to create reflective screen. Cheap alternative to plexiglass/acrylics

Appendix D
Trade Studies

MCU

	Particle Photon	WeAct Black Pill V2.0	Adafruit Feather	WeAct Black Pill V1.2	STM32F446	Dragonfly
Processor	STM32F205RGY6	STM32F411CEU6	STM32F405	STM32F401CCU6	STM32F446ME	STM32L496
Speed	120Mhz	100Mhz	168Mhz	84Mhz	180Mhz	80Mhz
Flash Size	1mb	512kb	1024kb	256 kB	512kb	1024kb + 16MB
RAM Size	128kb	128kb	128kb+64kb	64 kB	128kb	320kb
# of ADC channels	8 chan	10 chan	6 chan	16 chan	24 chan	3x16=48 chan
ADC Details	12bit/6MSPS/0.5us	12bit/2.5MSPS	12 bit, 2.4MSPS	12bit/2.5MSPS	theoretical 7.2MS	12-16bit, 5MSPS
# of SPI	2	5	2	1	4	3
DFSDM	No	No	No	No	No	Yes, 6 chan
SAI	None	None	None	None		2x, 4 lines exposed
FPU	None	Exists	Exists	Exists	Exists	Exists
Power Consumption	Typically 30uA of current					
Battery Support	Yes	No	Yes	No	No	Yes

Microphone

Model	ST MP34DT06J	ST MP34DT01M (on the Adafruit Board)	Adafruit Microphone Using I2S
Signal to Noise Ratio	64 dB	61 dB	65 db(A)
Output Type	PDM	PDM	I2S
Sensitivity	-26 dBFS ± 1 dB	-26 dBFS	-26 dBFS
Is it on a board	No	Yes	Yes
Current (Typ)	650uA	0.6mA	0.6mA

Battery

Model	TinyCircuits ASR0000	Lithium Polymer Battery	Adafruit 500mAh LIPO
Voltage	3.7V	3.7V	3.7V
Capacity	290mAh	1200mAh	500mAh
Weight	Unknown	23g	10.5g
Connector	JST SH Connector	JST PH Connector	JST PH Connector
Cost	$5.95	$9.95	$7.95