# Data collection

## Materials

* Raspberry Pi 3+
* Respeaker 6-Mic circular array
* Half circle protractor
* Portable speaker
* Smartphone with a sine signal generator app
* Digital caliper with 0.01 mm accuracy
* Laptop with Python 3 installed

## Procedure

### Sampling

Using Raspberry Pi 3+ with Respeaker 6-microphone circular array we have recorded six second audio samples of three different types of sounds: constant 432 Hz sinewave generated with a smartphone and played back through a portable speaker, speech, and claps both produced by one of the project members. We have sampled data from an uncertain distance further than two meters from the microphone setup, and from a range of orientations between 0° and 330° with steps of 30°. All recording were sampled with a 48000 Hz sampling frequency. We have collected three samples for each sound for each orientation segment. We have sampled data in a reverberant room at TechHub Assen (Image in the presentation) and in an acoustically treated room at Prince Claus Conservatorium Groningen to observe the effect of acoustics of the environment on the results.

We have decided to use sine, speech and clap sounds because of their distinct properties: sinewaves have a sustained and consistent tonal nature; claps have sharp transients with noise features; speech combines both sustained, transient, tonal, and noisy characteristics. Figure X, Y and Z show spectrograms of a random sample that we collected for each sound source for both reverberant and acoustically treated environments. 432 Hz sinewave was used because it has a wavelength of ~0.79 meters ensuring that the signal was actuated from a far field region.

### Pre-processing

Low frequencies can influence the accuracy of the system since their wavelength is longer and resulting in much larger near field region. We can observe from spectrograms that audio samples include low frequency content. In order to eliminate low frequencies from the audio samples we pre-process the audio samples before degree of arrival (DOA) estimation with a fifth order Butterworth high-pass filter with a cutoff frequency of 400 Hz. This configuration ensures that the frequencies that may be in the near field have a negligible magnitude comparing to the far field frequencies.

### MUSIC DOA estimation

Pyroomacoustics Python module was used for estimating DOA with MUSIC algorithm. To use MUSIC algorithm, the signal was sampled with a short-time Fourier transform with a window length of 24000 samples. MUSIC algorithm requires positional coordinates of the microphones which we have measured with a digital caliper. The coordinates relative to the center of the microphone array are shown in Table X.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Mic 1 | Mic 2 | Mic 3 | Mic 4 | Mic 5 | Mic 6 |
| X axis (meters) | 0.0230 | -0.0230 | -0.0460 | -0.0230 | 0.0230 | 0.0460 |
| Y axis (meters) | -0.0400 | -0.0400 | 0.0000 | 0.0400 | 0.0400 | 0.0000 |

Table X: measured microphone coordinates relative to the center of the microphone array.

The output of the algorithm included the DOA in radians which we transformed to degrees.

### GCC-PHAT DOA estimation

In order to provide a comparison to the results of MUSIC algorithm, we have created a GCC-PHAT based script for DOA estimation. Due to time constraints, we have decided to limit the system to three microphones. We have designed a GCC-PHAT algorithm which utilizes Mic 1, Mic 3, Mic 5 of the microphone array. In order to design the software, we have calculated the distances of the microphones from the center of the microphone using properties of an equilateral triangle with inscribed and circumscribed circles. We have implemented a function for estimating a line according to the GCC-PHAT output for each microphone pair and geometric values calculated earlier. Next, we have calculated a line-line intersection for each pair of lines. The output of the system was orientation of the mean of intersection points.