RPi Experiments:

Configuration:  
Brief introduction to raspberry Pi. Its current range of models.  
More detailed look at RPi3  
Soundcard we have chosen and other hardware.

Brief introduction to Python. Why we use it. What’s good.

Configuring the modules used for signal manipulation and analysis.   
PyAudio  
Numpy  
SciPy  
MatPlotLib  
SoundDevice  
Soundfile

<https://python-sounddevice.readthedocs.io/en/0.3.8/index.html>  
<http://pysoundfile.readthedocs.io/en/0.9.0/>

Audio Processing  
Playing audio can be done in many ways. The most straightforward is to create an array with audio data, and use the SoundDevice (sd) library’s play() function along with the sample rate:

sd**.**play(myarray, samplerate)

This will play the audio to the default device, unless specified.   
A file can be loaded from disk and stored in an array with SoundFile (sf) library’s read() function:

myarray, samplerate **=** sf**.**read(‘filename.wav’)

sd**.**play(myarray, samplerate)

This method will also read the audio file’s sample rate.

Recording audio can also be done using SoundDevice using the rec() function:

duration **=** 10.5 *# seconds*

myrecording **=** sd**.**rec(int(duration **\*** fs), samplerate**=**fs, channels**=**2)

This will record audio from the default input device for the duration specified and store it as the Numpy array myrecording

These basic methods are useful for simple applications, but require the entire audio file being played to be stored in memory. For very large files this can use all available resources

Streams

Audio Input

Input and output simultaneously: Wire, transceiver.

Signal generation: Tone, Multiple tones/chords, Stereo signals

Correlation: Cross-correlation, Autocorrelation, noisy correlation

DFT: Single tone, multiple tones, phase, spectral spread, zero-pad, window

TOA: send to receive.

TDOA: loopback, send to receive

FM Broadcast

ZC signals?  
Sync server?