Digital Health 2021/22
for
Students

Sonar App

by David Kopyto

Date: November 4, 2021
Supervisors: David Kopyto, Dr. Luis Lopera, Prof. Dr. Oliver Amft

1 Sonar: Principles

A Sonar is a Radar system that works with ultrasound instead of electromagnetic waves. Sonar systems emit an audio signal that is reflected at a target. The reflected signal is then recorded by a receiver (microphone). Depending on how the target moves, a Doppler shift is caused to the reflected signal. With that, we can estimate the motion of a user. There are several kinds of Sonar systems. Some send short pulses that are reflected at a target, some send a frequency modulated continuous wave (FMCW), and others just send one frequency continuously and record the Doppler shift for this frequency (continuous wave Sonar, CW). In this exercise, we want to build a CW Sonar that can track basic motions of the user (e.g. sleeping behavior) using a $20kHz$ ultrasound signal.

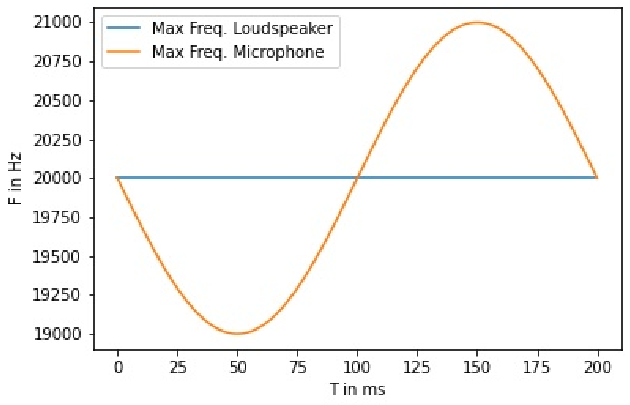

Figure 1: Sketch of expected maximum frequencies of loudspeaker and microphone in a Sonar setting.

2 Motion/Sleep Tracking

The basic Sonar functionality was already implemented in Exercise 2. We run the loudspeakers with a $20kHz$ ultrasound tone and record the audio simultaneously with the built-in Microphone. We expect to get slight frequency shifts when we record the maximum frequency that occurs at the microphone due to Doppler shifts caused by moving targets. This is schematically shown in Figure 1. What still needs to be figured out is how to use the recorded data for motion tracking. For that, we now design a basic signal processing pipeline which makes this possible.

2.1 Highpass Filter

We now take a deeper look at the scidart package. This package implements basic signal processing algorithms such as filters or FFT directly in Flutter. First take a look at the reference: `https://pub.dev/documentation/scidart/latest/` . First, we want to filter out frequencies outside the region where possible Doppler shifts caused by motion can lie. Theoretically, they can be anywhere in the audio spectrum, but as we want the system to work in the ultrasound range, we restrict it to work in very high frequencies which are inaudible for most people. We design a highpass filter with a cut-off frequency $f_c = 18kHz$. This suppresses all frequencies that are below f_c. Theoretically, we can then also detect motion-induced Doppler when e.g. music is played. To ensure stability, we use an FIR filter design. Scidart provides functionality for filter design. Please take a look at the functions firwin() and lfilter() in the scidart documentation. With firwin() we can compute the coefficients of the FIR filter. We need to set the variable pass_zero to false to create a highpass. Furthermore, we need to set the variable cutoff to $f_c = 18kHz$ and the sampling frequency fs to $44.1kHz$.

2.2 Fast Fourier Transform (FFT) of Filtered Data

Doppler shift means that the frequency of the test tone we send out with the loudspeakers is shifted by a certain amount by the motion of the reflecting target. To analyze this effect, we transform the filtered data into the frequency domain. This can be done using the provided FFT from scidart. Please take a look at the function fft() in the scidart reference.

```
var x = arrayToComplexArray(Array(listDouble));
var fftX = fft(x, n: 8192);
```

Here, we compute an FFT with size 8192 from the filtered microphone data listDouble. We need to cast the list to the scidart data type Array to compute it. The whole FFT computation is provided in fftSamples() in the code.

2.3 Motion/Sleep Tracking from Frequency Array

After getting the FFT values as a list, the motion shall be tracked. This is done by a simple decision rule. We look at the frequency band from $f_c = 18kHz$ to the Nyquist frequency ($f_s/2 = 22.05kHz$) and take the maximum frequency that occurs in this band. If there is no motion, no Doppler shift happened. That means the maximum frequency will be equal to $20kHz$, which is the center frequency of the Sonar. If a Doppler shift occurred, the maximum frequency will be changed. Then the CW signal was reflected by the target that was moving relative to the smartphone. To decide whether this motion is in the range of sleeping motion (small motions of the stomach etc.) or more heavy motions from getting up, we set a threshold for the maximum deviation from the center frequency.

```
bool isSleeping(List<int> fftValues) {
  var N = fftValues.length;
  var fs = 44100;
  var freqs = fftFreq(N, d: 1 / fs, realFrequenciesOnly: false);
  var centerFreq = 20000;
  var listDouble = fftValues.map((i) => i.toDouble()).toList();
  var argMaxFFT = arrayArgMax(Array(listDouble));
  var maxFFT = freqs[argMaxFFT];
  var diff = maxFFT - centerFreq;
  print(diff);
  return (diff.abs() <= 0.05 * centerFreq);
}
```

3 Questions and Exercises

1. Look at the function hpFir() in the code. Set up the calculation of coefficients using firwin() as explained.
2. Use lfilter with the coefficients that firwin returns. You can find a similar example in the scidart documentation. Which filter design method is used by default? (window function?)
3. Which filter order do you suggest (numtaps)? Play around with the numbers and check the results.
4. Experiment with different FFT sizes. Does that improve results?
5. Experiment with the threshold for sleeping detection. Which threshold would be a good fit?
6. The algorithm presented here was only using basic signal processing techniques such as filtering, transform, and thresholding. Can you think of a more advanced pipeline that would make results more promising for sleep motion detection?
7. Which other functionalities could enhance user experience in the Sonar App?