- Audio-Basics using WAV Format -

Goal:
With this exercise you learn fundamentals of one important Automotive application domain: Digital Audio; in particular you work on gernerating digital, uncompressed Audio.

Exercise 7.1: Preparation
First you check the audio functionality of your target system. You need keyboard, mouse and display to enter the menu line → speaker symbol → setup → test. Make sure you have speaker or headphones connected to the target.
Now you verify / install the following pakcets on the host and on the target:

- **`libsndfile1, libsndfile1-dev, sndfile-programs, hexer`**

As a quickstarter, pls. use the example program **`wav_writer.cc`** which can be found on http://webuser.hs-furtwangen.de/~coe. It is written in C++, since the C++-API of the sndfile library is much easier to handle. Now create a Managed Make project (C++) in Eclipse which we use for this exercise.

Exercise 7.2: Our first sine wave
Analyze the program and understand what it is doing. For this you might consult the documentation of the **`sndfile`** library. Now build the program for the host and execute it inside Eclipse. You need to tell Eclipse about the required library. Where do we find the generated WAV file? Using the hex editor, verify that the generated file is a correct WAV file. Playback the WAV file on the **host** using a suitable audio player, e.g. **`aplay`** or **`sudo sndfile-play`**.
For the **target**: copy the WAV-file using **`scp`** and play back.
Note: sometimes, at first the correct audio sink has to be activated on the target to get sound output. This is done by: **`sudo amixer -c 0 cset numid=3 1`**
You can control the volume using e.g. **`amixer sset 'PCM' 90%`**

Exercise 7.3: We can do even more - dynamics
Now copy your audio project into a new Eclipse project. Extend the **`wav_writer.cc`** by adding some dynamics. This means the volume of the sine tone is slowly increasing and decreasing and so on.
Hint: use an additional factor (float), which is combined with each of your samples. This factor also follows a sine wave, e.g. with a period of 1Hz.

Exercise 7.4: Almost an audio professional – 2 channel stereo sound
Again, copy your Eclipse projects. Extend the original code by a 2nd channel. What does this mean for the buffer size? How can we check, that we are really listening to stereo?
Hint 1: put another sine wave on the 2nd channel, which has a slightly different frequency than the 1st channel.
Hint 2 (optional): you can increase the stereo impression by oscillating the volume of each channel (see exercise 7.3)
Hint 3 (double bonus): you can increase the stereo impression even further by adding a small phase offset to the 2nd channel.