Homework #1 Report

Group #18
Eduard Ciprian Ilas s277901
Gabriele Cuni s277957
Valeria Sorrenti s276146

1. **Sensor Fusion Dataset with TFRecord**

   The first exercise aims to create a `TFRecord` dataset containing datatime, temperature and humidity values, together with serialized audio samples. In order to minimise the storage requirements of the dataset, we selected the `Int64List` format to store the datatime as posix timestamp, as well as the temperature and humidity values. Since the raw audio data can be simply represented as a bytestring, we decided to serialize it using the `BytesList` format. Using 1-second 48kHz 16bit audio samples, the final `TFRecord` size required 96152 bytes per record.

2. **Low-power Data Collection and Pre-processing**

   The second exercise required us to write a Python script in order to iteratively record 1-second audio samples and processing them. The processing phase would have to resample the audio to 16kHz, compute the STFT and then the MFCCs representation of the sample, which is then to be saved on disk. In order to limit the power consumption, we had to manually switch the board to *performance* mode only for short periods of time throughout each iteration, all while abiding to the timing constraint of having the processing phase lasting less than 80ms.

   We approached the requirements by identifying which were the operations that mostly would benefit from a boost in *performance*, and we found out that the most expensive operations were the spectrogram computation (130ms), followed by the MFCC computation (32ms). Regarding the recording phase, most of it was just idle time and would not benefit from a *performance* mode, although we made sure not to save the audio file on disk at the end of the sampling. Switching the governor was not instant and instead required several milliseconds, meaning we couldn't afford to switch to *performance* mode right before the expensive operations, since it would not be done by the time it was needed. Instead we found that the best moment to initiate the switch was during the idle times of the recording phase, specifically during the second-last chunk-reading iteration, while the switch to *powersave* mode was done at the end of each loop. To further increase efficiency, we computed the matrix used to map linear scale spectrogram to the Mel scale beforehand, since it was constant.

   By adopting this measures, we were able to lower the processing latency to around 65ms, and the `time_in_state` monitors reported around 460 time units in *powersave* mode.