Objective: The purpose of Part 1 of the lab is to obtain experience with the behavior of a timing-dependent system with and without explicitly defined deadlines, and under varying amounts of system load. First, you will design a tone generator task that produces the waveform of a tone with a given frequency (Step 1). Second, you will add a background task that will compete with the tone generator task in a way that will distort the tone in an audible way even during normal load conditions (Step 2). Third, you will run the tone generator task and background task together in a way that does not distort the generated tone under normal load conditions (Step 3). Finally, you will measure the execution time of the tone generator task and the background task for some given scenarios (Step 4).

Approval: When you see the text "*Assistant's approval: .....*" below a problem description you should show your solutions to the laboratory assistant. If the solutions are found to be satisfactory the laboratory assistant will sign your lab-PM and mark the corresponding examination objective as 'Passed' in Canvas. You can then continue with the next problem.

Step 1: And then there was sound ...

Implement a pulse wave tone generator on card MD407, by means of a periodic task that feeds the card's DAC (digital-to-analog converter) with alternating '1's and '0's. In this context a '0' means writing the value 0 to the DAC, whereas a '1' means writing a value between 1 and 20 (0x14 in hexadecimal notation) to the DAC. Set the tone frequency to 1 kHz. The memory address of the 8-bit DAC data register is 0x4000741C. If implemented correctly, the pulse wave should produce an audible output in a speaker connected to the audio jack on the MD407 card.

CAUTION: Because the output of the DAC is connected to a speaker output without any attenuation you should never write values larger than 20 to the DAC. Otherwise, the output signal may ruin your headphone speakers and/or your hearing. Recommended values, producing a comfortable sound level on most speakers, are in the range 5–15.

You should now implement support for controlling the tone generator task from your keyboard, without stopping the periodic execution of the tone generator task. First, implement a *volume control* that allows you to increase and decrease the value, representing a '1', to the DAC. Make sure that your volume control always keeps the value within the minimum and maximum limits of 1 and 20, respectively. Control the volume up/down function with suitable keys on the keyboard. Second, implement a *mute function* that allows you to enable and disable the sound on your card's audio jack. Control the mute/unmute function with suitable keys on the keyboard.

> **Problem 1:** Demonstrate the audible 1 kHz output, as well as the volume control and mute function, to the teaching assistant.

Assistant's approval: .....

Step 2: Fifty shades of distortion

Extend the program from Step 1 with a periodic background load task with a period of 1300 μs, that, for each invocation (each new period), executes an empty loop a number of times defined by a variable labeled `background_loop_range`. The intention is that the background task should increase the overall load of the processor in a controlled manner, where larger values of the variable `background_loop_range` correspond to higher load, and smaller values correspond to lower load.

The background load task should be run concurrently with the tone generator task from Step 1. Remember that, in order to implement two concurrent tasks in TinyTimber, you need to use two independent objects. Place the variable `background_loop_range` inside the object you use for the background task, and give it a default value of 1000. If implemented correctly, you should hear a "dirtier" version of the 1 kHz tone that is being distorted by the periodic executions of the background task.

You should now implement a *load control* that allows you use the keyboard to increase and decrease the value of variable `background_loop_range` while both tasks are running. It is recommended that you increment and decrement in steps of 500. Control the load increase/decrease function with suitable keys on the keyboard. Print the value of variable `background_loop_range` every time it is changed.

Problem 2.a: With the tone generator producing a 1 kHz tone increase the value of variable `background_loop_range` from 1000 to 8000. What happens with the generated tone as you increase the background load?

Answer: .....

Problem 2.b: Repeat the procedure in Problem 2.a, but let the tone generator task produce a 769 Hz tone. Is the distortion different now? If so, why? What happens with the generated tone as you increase the background load?

Answer: .....

Problem 2.c: Repeat the procedure in Problem 2.a, but let the tone generator task produce a 537 Hz tone. Is the distortion different now? If so, why? What happens with the generated tone as you increase the background load?

Answer: .....

Assistant's approval: .....

Step 3: Deadlines to the rescue

Add an explicit deadline of 100 μs to the 1 kHz tone generator task. Also assign a deadline of 1300 μs to the background load task (that is, equal to its period). The intention is to make use of the timing-aware scheduling in the TinyTimber kernel, and demonstrate its usefulness in an application with strict timing constraints. Give the variable `background_loop_range` a default value of 1000. If implemented correctly, you should now once again hear the clean sounding tone from Step 1.

Problem 3.a: Add a *deadline control* that allows you to enable and disable the deadlines of both tasks simultaneously. Control the deadline enable/disable function with suitable keys on the keyboard.

Problem 3.b: With deadlines enabled for the two tasks, and the tone generator producing a 1 kHz tone, use the load control keys to increase the value of variable `background_loop_range` from 1000 to 8000. What happens with the generated tone as you change the background load?

Answer: .....

Problem 3.c: With deadlines enabled for the two tasks, and the tone generator producing a 1 kHz tone, increase the background load beyond the range used in Problem 3.b until you hear a *pitch drop effect* where the tone generator starts producing a tone with a lower frequency. What is the value of variable `background_loop_range` when the pitch drop effect begins to be noticeable?

Answer: .....

Assistant's approval: .....

Step 4: What's the time?

You should now measure the worst-case execution time (WCET) of the program code in the tone generator task and the background task, respectively. The execution time of a task may vary from one execution to another (for example, due to the variation in hardware or software behavior). Therefore, you should measure the WCET of a task by running it 500 times, and then derive the *maximum* WCET as well as the *average* WCET based on data that you collect during these executions. To get as accurate estimates of the WCET as possible, do not include any ASYNC/AFTER/SEND calls in the measurement.

Problem 4.a: Measure the WCET of the background task by running it in isolation, that is, without running the tone generator task. The variable `background_loop_range` should have the value that you identified in Problem 3.c for the pitch drop effect. Perform the measurement by running the task a large number of times, as described above. What are the values for the maximum WCET and average WCET, respectively?

Answer: .....

Problem 4.b: Measure the WCET of the background task by running it in isolation, that is, without running the tone generator task. The variable `background_loop_range` should have a value of 1000. Perform the measurement by running the task a large number of times, as described above. What are the values for the maximum WCET and average WCET, respectively?

Answer: .....

Problem 4.c: Measure the WCET of the tone generator task by running it in isolation, that is, without running the background task. Perform the measurement by running the task a large number of times, as described above. What are the values for the maximum WCET and average WCET, respectively?

Answer: .....

Assistant's approval: .....

That's it for Part 1.
Have fun with Part 2!