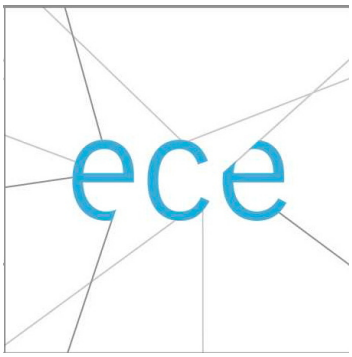
EXTENDING NANO-RK:
EDF SUPPORT



Name	Student Number	Email Address
Oscar Lee	16614075	oscarkmlee@gmail.com
Nick Adams	61406088	nick@nickadams.ca
Catherine Wang	64746084	moonhacesol@gmail.com
Jack Wu	17254079	jack.nan.wu@gmail.com

EECE 494
Department of Electrical and Computer Engineering
University of British Columbia

Modified File
/nano-RK/src/kernel/source/nrk_task.c : nrk_add_to_readyQ(int8_t taskID)

Nano-RK Scheduling

To ensure the tasks are scheduled according to earliest deadline first, the algorithm for inserting a task in the ready queue is modified. Since the schedule class execute tasks in the ready queue sequentially, from the head of the linked list to the tail, how the tasks are inserted to the ready queue determines which scheduling policy is used by the scheduler.

Originally, in function nrk_add_to_readyQ, the tasks were inserted into the ready queue based on preset task priorities. That is, the function searched for the first task that has a lower priority than the *new task*, by traversing the doubly-linked list data structure. Afterwards, the *new task* is inserted into the queue*,* via it's task ID, in front of this task. Resulting in the ready queue ordered in decreasing priority.

Nano-RK Scheduling with EDF
After our modification, instead of using the priority metric, the function now searches for the first task with a deadline later than the *new task*, and inserts the task in front similar to the original approach. The correct metric to use to compare relative deadlines in this case is the wakeup_time member, since it is a decreasing counter which is set to the period of each task as an indication for when the task should be ran again next.

The following logic is used for each task in the doubly-linked queue:

1. *Is next task an idle task? if true, insert the new task in front of this idle task. Given that we reached this point already, tasks in front of the idle task must all have a shorter deadline and have to be executed first.*
2. *If both the new task and the current task being compared are scheduled to execute now (i.e. wakeup_time = 0), then their periods would be compared and the task with the shorter period will be scheduled ahead.*
3. *Else if the new task is scheduled to execute now, then it should be placed in front of the first task in the ready queue it finds that has a wakeup_time != 0.*
4. *Else if both the new task and current task being compared are not scheduled to execute (i.e. wakeup_time != 0), then only their wakeup times would be compared and the task with the earlier wake up time will be scheduled ahead.*

Note: The relative deadline of each task is assumed to be equal to its period.

Test Verification
We verified the correctness of our EDF implementation by running some basic scheduling tests to verify the correct order of the task scheduling. Using a task set with three tasks we are able to verify correct behavior for task scheduling in three cases, inserting at beginning of queue, inserting at end of queue, and inserting somewhere in the middle of the queue. The following task set was used in verifying nano-RK with earliest deadline first.

Task ID	Period	Execution Time
1	8	4
2	10	2

3	15	3

This task set allowed us to confirm EDF by adding at the beginning with the first task, in the middle with the second task and at the end with the third task. We also verified the order of task execution.