

What is claimed is:

1. A method of scheduling tasks for a Real-Time Operating System (RTOS) in a low-power, wireless, mesh network, the method comprising:
receiving, at a scheduler for the RTOS, a first plurality of tasks to schedule for execution by one or more processors and a second plurality of tasks for execution by the one or more processors, wherein:
each task in the first plurality of tasks is:
associated with an expected execution interval that indicates an expected length of time for the one or more processors to execute the task;
associated with a list of resource identifiers that identify resources required to execute each of the first plurality of tasks;
and associated with an execution time that indicates when the task should begin execution by the one or more processors; and
each task in the second plurality of tasks is:
associated with an expected execution interval that indicates an expected length of time for the one or more processors to execute the task;
associated with a list of resource identifiers that identify resources required to execute each of the second plurality of tasks; and
not associated with an execution time that indicates when the task should begin or end execution by the one or more processors, such that the task can be scheduled at any time when the expected execution interval is available at the one or more processors;
scheduling the first plurality of tasks for execution by the one or more processors such that each of the first plurality of tasks is scheduled to begin at the task's respective execution time;
scheduling the second plurality of tasks for execution by the one or more processors such that each of the second plurality of tasks is scheduled in a time interval left after scheduling the first plurality of tasks that is at least as long the task's associated expected execution interval; receiving a walk-in task that has a list of resource identifiers that identify resources required for execution of the walk-in task, wherein the list of resources comprises a reference to at least one mutually exclusive section of code;
searching the list of resource identifiers for each task in the first plurality of tasks for

references to the at least one mutually exclusive section of code;
searching the list of resource identifiers for each task in the second plurality of tasks for references to the at least one mutually exclusive section of code; and
scheduling the walk-in task for execution by the one or more processors such that:
the walk-in task can be preempted by ones of the first plurality of tasks and ones of the second plurality of tasks that do not include a reference to the at least one mutually exclusive section code in their respective lists of resource identifiers; and
the walk-in task is not preempted by ones of the first plurality of tasks and ones of the second plurality of tasks that do include a reference to the at least one mutually exclusive section code in their respective lists of resource identifiers.

2. The method of claim 1, further comprising:
determining that a task in the first plurality of tasks cannot be scheduled to begin execution at the task's execution time; and
executing a callback function to provide a notification that the task cannot be scheduled to begin execution at the task's execution time.

3. The method of claim 1, wherein:
each of the first plurality of tasks is associated with respective priorities;
each of the second plurality of tasks is associated with respective priorities;
the walk-in task is associated with a priority;
each of the second plurality of tasks are scheduled for execution by the one or more processors according to their respective priorities;
each of the second plurality of tasks are scheduled for execution by the one or more processors according to their respective priorities;
the walk-in task is scheduled according to its priority; and
scheduling the walk-in task causes one of the first plurality of tasks to miss its associated execution time.

4. The method of claim 1, wherein:
a first task is associated with a priority;
the priority indicates a scheduling priority relative to other tasks; and
the first task is scheduled using the priority.

5. The method of claim 1, wherein the expected execution interval of one of the first plurality of tasks is provided to the scheduler by an application associated with the one of the first plurality of tasks.

6. The method of claim 1, wherein:
the RTOS allows preemption;
a first task declares a resource identifier for a first resource;
and the first task is scheduled to use the first resource based on the first task's expected execution time.

7. A low-power wireless sensor device running a Real-Time Operating System (RTOS), the sensor device comprising:
a wireless communication module;
one or more processors; and
a memory having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations including:
receiving, at a scheduler for the RTOS, a first plurality of tasks to schedule for execution by one or more processors and a second plurality of tasks for execution by the one or more processors, wherein:
each task in the first plurality of tasks is:
associated with an expected execution interval that indicates an expected length of time for the one or more processors to execute the task;
associated with a list of resource identifiers that identify resources required to execute each of the first plurality of tasks; and associated with an execution time that indicates when the task should begin execution by the one or more processors;
and each task in the second plurality of tasks is:
associated with an expected execution interval that indicates an expected length of time for the one or more processors to execute the task;
associated with a list of resource identifiers that identify resources required to execute each of the second plurality of tasks; and
not associated with an execution time that indicates when the task should begin or end execution by the one or more processors, such that the task can be scheduled at any time when the expected execution interval is available at the one or more processors;

scheduling the first plurality of tasks for execution by the one or more processors such that each of the first plurality of tasks is scheduled to begin at the task's respective execution time;

scheduling the second plurality of tasks for execution by the one or more processors such that each of the second plurality of tasks is scheduled in a time interval left after scheduling the first plurality of tasks that is at least as long the task's associated expected execution interval;

receiving a walk-in task that has a list of resource identifiers that identify resources required for execution of the walk-in task, wherein the list of resources comprises a reference to at least one mutually exclusive section of code;

searching the list of resource identifiers for each task in the first plurality of tasks for references to the at least one mutually exclusive section of code; searching the list of resource identifiers for each task in the second plurality of tasks for references to the at least one mutually exclusive section of code; and

scheduling the walk-in task for execution by the one or more processors such that: the walk-in task can be preempted by ones of the first plurality of tasks and ones of the second plurality of tasks that do not include a reference to the at least one mutually exclusive section code in their respective lists of resource identifiers; and the walk-in task is not preempted by ones of the first plurality of tasks and ones of the second plurality of tasks that do include a reference to the at least one mutually exclusive section code in their respective lists of resource identifiers.

8. The device of claim 7, wherein the instructions cause the one or more processors to perform further operations including:

determining that the a task in the first plurality of tasks cannot be scheduled to begin execution at the task's execution time; and

executing a callback function to provide a notification that the task cannot be scheduled to begin execution at the task's execution time.

9. The device of claim 7, wherein:

each of the first plurality of tasks is associated with respective priorities;

each of the second plurality of tasks is associated with respective priorities;

the walk-in task is associated with a priority;

each of the second plurality of tasks are scheduled for execution by the one or more

processors according to their respective priorities;
each of the second plurality of tasks are scheduled for execution by the one or more processors according to their respective priorities;
the walk-in task is scheduled according to its priority; and scheduling the walk-in task causes one of the first plurality of tasks to miss its associated execution time.

10. The device of claim 7, wherein:

a first task is associated with a priority; the priority indicates a scheduling priority relative to other tasks; and
the first task is scheduled using the priority.

11. The device of claim 7, wherein the expected execution interval of one of the first plurality of tasks is provided to the scheduler by an application associated with the one of the first plurality of tasks.

12. The device of claim 7, wherein:

the RTOS allows preemption; a first task declares a resource identifier for a first resource; and
the first task is scheduled to use the first resource based on the first task's expected execution time.

13. A non-transitory, computer-readable storage medium having instructions stored thereon which, when executed by one or more processors, cause the one or more processors to perform operations including:

receiving, at a scheduler for the RTOS, a first plurality of tasks to schedule for execution by one or more processors and a second plurality of tasks for execution by the one or more processors, wherein:

each task in the first plurality of tasks is:

associated with an expected execution interval that indicates an expected length of time for the one or more processors to execute the task;

associated with a list of resource identifiers that identify resources required to execute each of the first plurality of tasks; and

associated with an execution time that indicates when the task should begin execution by the one or more processors; and

each task in the second plurality of tasks is:
 associated with an expected execution interval that indicates an expected length of time for the one or more processors to execute the task;
 associated with a list of resource identifiers that identify resources required to execute each of the second plurality of tasks; and
 not associated with an execution time that indicates when the task should begin or end execution by the one or more processors, such that the task can be scheduled at any time when the expected execution interval is available at the one or more processors;
 scheduling the first plurality of tasks for execution by the one or more processors such that each of the first plurality of tasks is scheduled to begin at the task's respective execution time;
 scheduling the second plurality of tasks for execution by the one or more processors such that each of the second plurality of tasks is scheduled in a time interval left after scheduling the first plurality of tasks that is at least as long the task's associated expected execution interval;
 receiving a walk-in task that has a list of resource identifiers that identify resources required for execution of the walk-in task, wherein the list of resources comprises a reference to at least one mutually exclusive section of code;
 searching the list of resource identifiers for each task in the first plurality of tasks for references to the at least one mutually exclusive section of code;
 searching the list of resource identifiers for each task in the second plurality of tasks for references to the at least one mutually exclusive section of code; and scheduling the walk-in task for execution by the one or more processors such that:
 the walk-in task can be preempted by ones of the first plurality of tasks and ones of the second plurality of tasks that do not include a reference to the at least one mutually exclusive section code in their respective lists of resource identifiers; and
 the walk-in task is not preempted by ones of the first plurality of tasks and ones of the second plurality of tasks that do include a reference to the at least one mutually exclusive section code in their respective lists of resource identifiers.

14. The non-transitory, computer-readable storage medium of claim 13 wherein:
 each of the first plurality of tasks is associated with respective priorities;
 each of the second plurality of tasks is associated with respective priorities;

the walk-in task is associated with a priority;
each of the second plurality of tasks are scheduled for execution by the one or more processors according to their respective priorities;
each of the second plurality of tasks are scheduled for execution by the one or more processors according to their respective priorities;
the walk-in task is scheduled according to its priority; and scheduling the walk-in task causes one of the first plurality of tasks to miss its associated execution time.

15. The method of claim 1, wherein the expected execution interval of one of the first plurality of tasks is estimated by the scheduler based on previous execution times of the one of the first plurality of tasks.

16. The device of claim 7, wherein the expected execution interval of one of the first plurality of tasks is estimated by the scheduler based on previous execution times of the one of the first plurality of tasks.

17. The non-transitory, computer-readable storage medium of claim 13, wherein the expected execution interval of one of the first plurality of tasks is estimated by the scheduler based on previous execution times of the one of the first plurality of tasks.

18. The non-transitory, computer-readable storage medium of claim 13, wherein the instructions cause the one or more processors to perform further operations including: determining that a task in the first plurality of tasks cannot be scheduled to begin execution at the task's execution time; and executing a callback function to provide a notification that the task cannot be scheduled to begin execution at the task's execution time.