# **Embedded Systems Essentials with Arm: Get Practical with Hardware**

## Module 4

## KV4: An RTOS Example

Now we are ready to take a look at an assessment of an Mbed RTOS application.

As an example, let’s first undertake an initial evaluation of a programming requirement in general RTOS terms, in other words, not necessarily applying the Mbed RTOS.

In the following scenario, we will consider aspects of tasks, priorities, scheduling, time-slicing, timing deadlines, use of resources, and inter-task communication.

Our objective is an experimental, light-seeking, wheeled robotic vehicle, that is controlled by a single microcontroller, and is to be programmed with an RTOS. The vehicle speed will not exceed 0.5 meters per second.

It needs to undertake these actions:​

1. Sense obstacles ahead with an ultrasound sensor, which may take up to 6 milliseconds. ​
2. Sense a chassis tilt from the horizontal, with a three-axis accelerometer.
3. Sense direction of the brightest light, by reading the light sensor array.
4. Compute and output PWM, or pulse width modulation, values to drive its DC motors. It must drive towards the light, trialing simple avoidance routes if obstacles or excessive tilt hinders progress.
5. Transmit status information, meaning data from all of the actions mentioned here, to a base station. This may take up to 10 milliseconds.

Knowing what we do from the previous lessons, let’s return to the aspects of this program and what approach must be taken for this specific scenario.

Tasks: A simple approach would be to allocate one task to each of the five activities listed. Let’s adopt this initially, retaining the possibility that we might merge two or three, if the opportunity arises. ​

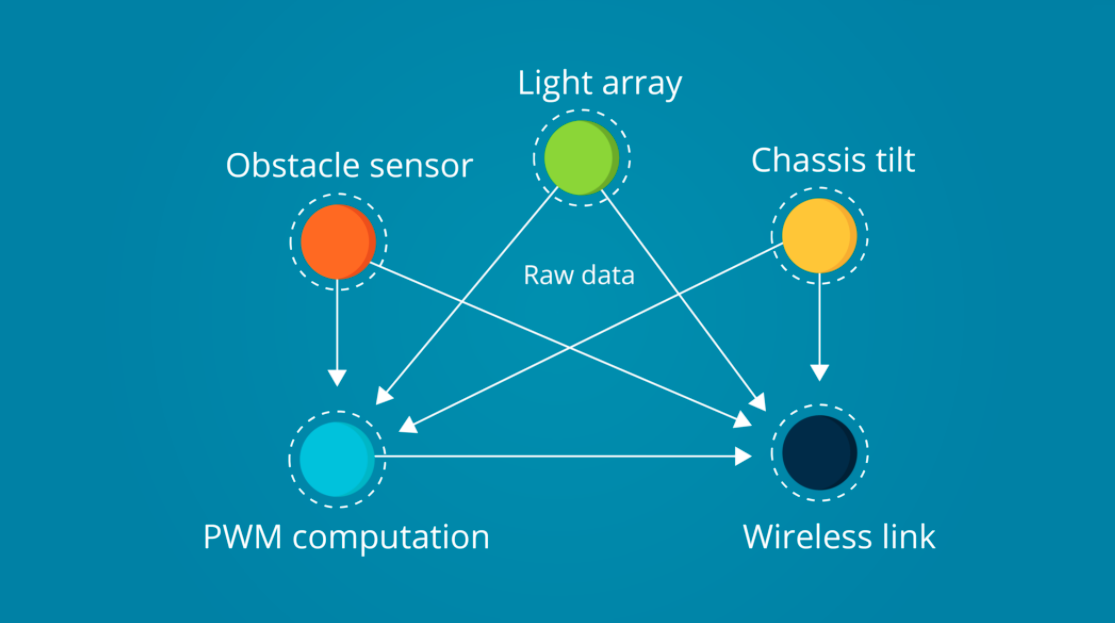
Priorities: Tasks appear interdependent. None dominate and none can be discounted. Therefore, it is proposed that they should be of equal priority. ​

Scheduling: Because all tasks are equal priority, a round robin or cooperative approach can suffice. A generous time-slice of 10 to 15 milliseconds, long by normal standards, would allow each task to complete within a time slice. ​

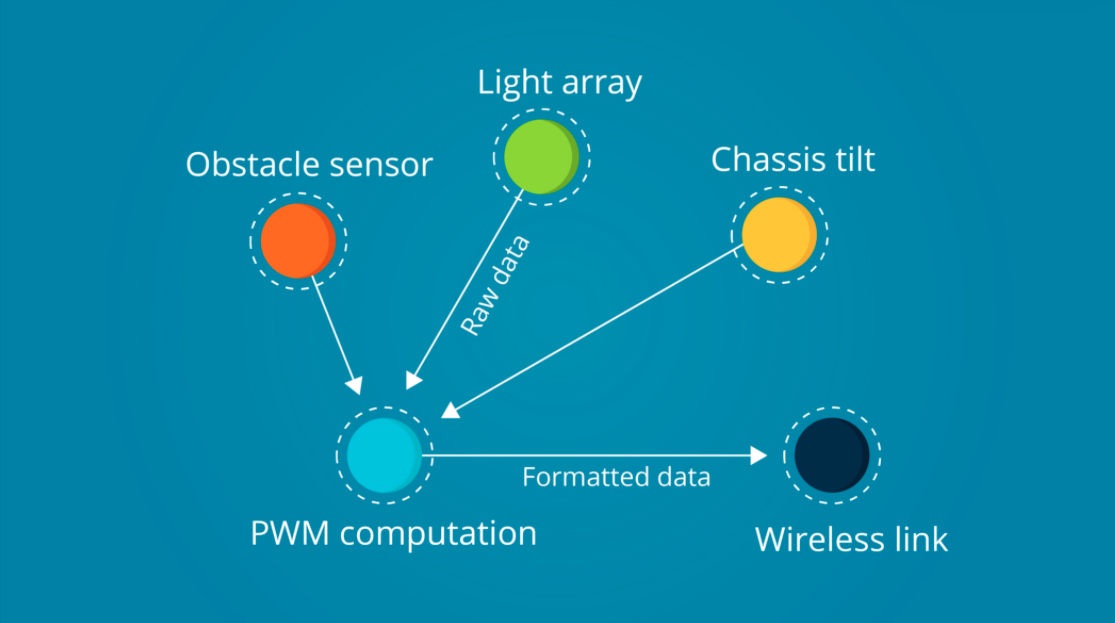
Timing Deadlines: Little timing information is given, but what there is allows some estimate of deadlines to be made. The vehicle moves rather slowly, and two tasks have time requirements in excess of 1 millisecond. With round robin scheduling, and a time slice of 15 milliseconds, each task would return every 75 milliseconds. During this time, the vehicle will move a maximum of 0.075 multiplied by 0.5 meters, which is less than 4 cm. It is likely that this will provide an adequate update rate for the PWM.​

Shared Resources: It would appear that there are no shared resources. However, depending on further detail being available, the wireless link could be viewed as one, and the program structured accordingly. In this case, appropriate resource control techniques, like mutex or semaphore, could be considered.​

With all these tasks and actions required, there are a fair amount of data transfers taking place, even though the overall quantity of data is limited. This leads us to inter-task communication, and how best to simplify.

This diagram shows the required raw data transfers from all the sources,

and this diagram shows how a reduction in data flow might be possible.



If the PWM computation task acted as the conduit for all sensor data to the wireless link, it could format the data more efficiently. Consideration can also be given to the use of a queue to manage data flow.

**Visit arm.com for more information.**