**Chapter 3-2 RTOS Semaphore**

Welcome back to WICED WiFi 101. Next I am going to talk about the semaphore. What is a semaphore? It is just a flag. But why? Well to answer that question I will paraphrase the 400-year-old poem and say “no thread is an island”. In general, threads need to communicate with each other… although the threads share the same memory address space, and as such can read and write each other’s variables. Doing so is inviting disaster or even worse very difficult to figure out bugs.

So to solve this problem the Dutch computer scientist [Dijkstra](https://en.wikipedia.org/wiki/Edsger_Dijkstra) invented the semphaore for threaded programming. All it does is allow you to send a signal from one thread to another…

Here is a flag that I like… the finish line judge at the race track flagging the Cypress car for the win.

Anyway. A perfect time to use a semaphore is inside of an interrupt service routine. This is a situation where you want to send a signal to a thread that it can move on.

In the WICED SDK we give you the ability to create semaphores using wiced\_rtos\_init\_semaphore. Make sure that you init the semaphore before you try to set or get it.

You can signal the semaphore, with wiced\_rtos\_set\_semaphore …

And finally wait for the semaphore using wiced\_rtos\_get\_semaphore. This function is really cool as it will put your thread to sleep until the semaphore has been set.

Unfortunately, there is no consensus in the world for set/get and you will see it as many other things including p/v take/get

In the textbook you will find this example under chapter 3 example 2 and in the answers as ww101key/03/02\_semaphore

For the semaphore example we ask you to use a button press to signal a thread to toggle an LED.

To build this program we will start in the application\_start function by initializing the semaphore and creating the button toggle thread, and finally turning on the interrupt.

Then I will make an interrupt service routine called “button\_isr” that will just set the semaphore when it is called.

And finally, the buttonThread will wait around for a semaphore, then, when it is set, will toggle the led.

Here we told the thread to wait forever until the semaphore is set. If you instead want the thread to stop waiting after some time and continue on even if the semaphore isn’t set then you would specify a timeout value in milliseconds.

This is a simple example, but the tool is very powerful. The bottom line is, when you need to send a thread safe message, use a semaphore.

As always, you can post your comments and questions in our Wifi developer community or you are welcome to email me at alan\_hawse@cypress.com or tweet me at @askioexpert with your comments, suggestions, criticisms and questions.

In the next video I will talk about the mutex