- Analysis:
  - Why `ulTaskNotifyTake` is not the correct notify API to use here:
  - The FreeRTOS Reference Manual (page 123), describes how `ulTaskNotifyTake` is just intended for use in implementing a fast semaphore:
    - *"ulTaskNotifyTake() is intended for use when a task notification is used as a faster and lighter weight alternative to a binary semaphore or a counting semaphore. FreeRTOS semaphores are taken using the xSemaphoreTake() API function, ulTaskNotifyTake() is the equivalent that uses a task notification value instead of a separate semaphore object..."*
    - *"When a task is using its notification value as a binary or counting semaphore other tasks and interrupts should send notifications to it using either the xTaskNotifyGive() macro, or the xTaskNotify() function with the function's eAction parameter set to eIncrement (the two are equivalent)."*
  - Using `ulTaskNotifyTake` adds some unnecessary complexity.
    - `ulTaskNotifyTake` returns the calling-task's notification-value, and it alters the notification-value when exiting. The parameter `xClearCountOnExit` specifies how the notification-value is altered on exit:

```
ulTaskNotifyTake( BaseType_t xClearCountOnExit, TickType_t xTicksToWait );
```

    - In `mainTaskNotifications.c`, the call is:

```
uint32_t notificationvalue = ulTaskNotifyTake( pdTRUE, portMAX_DELAY );
```

    - In using ulTaskNotifyTake here, the task's notification-value must be set to zero on exit. It doesn't need to be zero for the example-program's calculations. It needs to be zero to make `ulTaskNotifyTake` operate correctly, since it's designed to perform semaphore-like processing.
    - For example, if the task's notification-value is greater than zero when `ulTaskNotifyTake` exits, the next call to `ulTaskNotifyTake` can run whether or not there is a pending notification.
    - If `xTaskNotifyWait` is used instead, the task's notification-value on exit isn't a concern.

- **Bug** in FreeRTOS, in its documentation for `ulTaskNotifyTake`:
  - The book presents an example-program `mainTaskNotifications.c`, on page 227 As just described, above, there is a bug in the program's use of ulTaskNotifyTake.
  - The example-program has an additional apparent bug in its use of ulTaskNotifyTake. However, the bug is actually in the FreeRTOS documentation for ulTaskNotifyTake.
  - In the program, one of the calls to xTaskNotify sets the task-notification-value to 0:
    - `xTaskNotify( recvTaskHandle,0, eSetValueWithOverwrite);`
  - According to the API documentation, ulTaskNotifyTake will not return if the task's notification-value is 0. This is a documentation error, and I've reported it on FreeRTOS's Kernel forum.
    - https://forums.freertos.org/t/possible-bug-in-ultasknotifytake-it-can-return-incorrectly-when-the-notification-value-is-0/11774
  - When the example-program is run, ulTaskNotifyTake does return when the notification-value is 0.