```
// ..
#include <linux/delay.h>
#include <linux/workqueue.h>

static struct workqueue_struct *wq;
static void work_to_do(struct work_struct*);
// takes a function work_to_do and declares queable work_obj
static DECLARE_WORK(work_obj, work_to_do);
// declares completion object
static DECLARE_COMPLETION(on_exit);
static atomic_t stop_timer = ATOMIC_INIT(0);

static void work_to_do(struct work_struct *work) {
    // kernelkontext --> allowed to sleep
    msleep(2000);
    if (atomic_read(&stop_timer)) {
        complete(&on_exit);
        return;
    }
    if (queue_work(wq, &work_obj)) {
        printk("queue_work SUCCESS\n");
    } else {
        printk("queue_work ERROR\n");
    }
}

static int __init ModInit(void) {
    // ..
    // define wq with threadname
    wq = create_workqueue("Threadname");
    // add work_obj to work que
    if(queue_work(wq, &work_obj)) {
        printk("queue_work successful ...\n");
    } else {
        printk("queue_work not successful ...\n");
    }
    return 0;
}

static void __exit ModExit(void) {
    atomic_set(&stop_timer, 1);
    wait_for_completion(&on_exit);
    if(wq) {
        destroy_workqueue(wq);
        pr_debug("workqueue destroyed\n");
    }
    // ..
}
// ..
```